TESTBEDS AND EVALUATION FRAMEWORKS FOR ANOMALY DETECTION WITHIN BUILT ENVIRONMENTS: A SYSTEMATIC REVIEW

Mohammed Alosaimi
School of Computer Science and Informatics
Cardiff University, UK
alosaimimm1@cardiff.ac.uk

Omer Rana
School of Computer Science and Informatics
Cardiff University, UK
ranaof@cardiff.ac.uk

Charith Perera
School of Computer Science and Informatics
Cardiff University, UK
Pererac@cardiff.ac.uk

November 1, 2023

ABSTRACT

active in exchanging communication and data. IoT devices cannot enforce advanced security mechanisms to protect themselves from cyberattacks. Furthermore, IoT devices could behave abnormally when faulty or when a behavioral anomaly caused by surrounding factors is detected. Identifying what could be an anomaly is challenging because an anomaly might be unknown beforehand. Even though anomaly detection techniques are crucial to ensure the security of IoT devices, we must have a quality testbed capable of testing different anomaly detection techniques. The testbed is always an afterthought as effort often goes into anomaly detection techniques, neglecting the importance of the testbed. In this systematic review, we propose testbed characteristics in the context of anomaly detection and consider the testbed as a first-class citizen. We also discuss different anomaly detection techniques that were tested on an implemented testbed or a dataset, how anomalies are generated, and how anomaly detection techniques are evaluated. Dataset data creation, capture duration, and analysis are also discussed. Finally, we highlight helpful tools that are used in the context of building a testbed or creating a dataset that we came across in the literature.

***K*eywords** smart home, datasets, evaluation metric, network feature, smart building

1 Introduction

Internet of Things (IoT) refers to the concept of enabling devices, "things," to exchange data without human intervention [1]. An example of such a device would be a sensor, an actuator, a computer, or an object that can monitor or interact with the surrounding environment. Globally, IoT is estimated to have a $3.9 trillion to $11.1 trillion economic impact in 2025 [2]. Furthermore, it is expected that by 2025, there will be 27 billion IoT devices [3].

Because IoT technology is becoming increasingly popular, they have been vulnerable to cyber threats, including unauthorized access and spying on occupants via their cameras to hijacking voice assistants such as Alexa [4]. It is important to note that the distribution of IoT devices throughout the house poses a threat to the privacy and security of the data being read by these devices [5]. The integrity of the data could be compromised if an intruder gains access to the data and alters it. An intruder could, in a far worse scenario, have access to an IoT device that is responsible for opening and closing the front door and enters the house. As mentioned in this paper [6], an attacker can also determine whether a home is occupied by analyzing network traffic. Thus, it is imperative to detect anomalous behaviors to maintain the integrity and privacy of the data.

To identify a cyber or physical threat, a system should monitor the data and detect when data are abnormal or anomalous. According to Barnett and Lewis, "an outlier is an observation (or subset of observations) which appears to be inconsistent with the remainder of that set of data" [7]. In the context of IoT devices, Cook et al. defined an anomaly as "the measurable consequences of an unexpected change in the state of a system which is outside of its local or global norm" [8].

The process of detecting an abnormal behavior is complex. Because of the lack of a clear definition of what is abnormal, it is not as simple as detecting a static threshold [9]. A condition that is considered normal now may become abnormal in the future due to changes in the environment affecting factors such as the outside temperature. Additionally, it is possible that the observed data may show a trend or a seasonal change that would suggest an abnormal behavior, even though it is not abnormal behavior. On the contrary, when a person is found lying on his/her kitchen floor, it must be recognized as an anomaly because this is not part of the person's routine, and emergency services must be notified [10]. What is normal for one individual might not be normal for another. This example illustrates how anomaly detection can be challenging. Anomalies are, therefore, difficult to detect and define. Even though anomaly detection techniques are useful for detecting anomalies in IoT devices, most of them are tailored to a particular application [8].

Scientists conducted several anomaly detection literature reviews in the past. Our contribution is that we discuss anomaly detection algorithms in the context of the built environment and focus on the testbed. We consider it a first-class citizen because having a quality testbed would help standardize the evaluation process and evaluate different anomaly detection techniques thoroughly. We also discuss the metrics used to evaluate anomaly detection techniques, different types of anomalies, and anomaly creation. Furthermore, we discuss smart home datasets in terms of creating and collecting data, capture duration, and dataset analysis. We also highlighted helpful tools in that context.

This systematic review is divided into the following: Section 2 reexamines existing surveys and reviews that have been published in the same domain. Section 3 provides background information about the types of anomalies, the techniques to detect anomalies, and Reinforcement Learning (RL) in the context of anomaly detection. Section 4 discusses the research methods followed in this systematic review and our contribution, outlines research questions, and filters criteria. Section 5 investigates how testbeds are created according to several research papers, how an anomaly is created within the environment, and how testbed devices and participating users are selected; Section 6 explains different anomaly detection mechanisms in the literature along with their chosen anomaly detection algorithms; Section 7 discusses the infrastructure dataset setup, network configuration, data collection, data capture duration, and dataset analysis. Section 8 lists several practical tools in the context of building/generating a testbed/dataset. Section 9 reviews the most common anomaly detection evaluation metrics that we found in the literature.

2 Existing Surveys

Review and survey papers related to this study have been published, but their focus was on anomaly detection techniques, leaving out the testbed that was used to test such anomaly detection techniques. We also included papers from different fields of study, such as medical surveys, to investigate the issue of having the testbed as a second-class citizen. Furthermore, review and survey papers differ in methodology, the study domain, or the types of feedback loops considered. Using these reviews as a starting point, we highlight key aspects of our work and position our research in relation to them. Table 1 summarizes the existing surveys.

Table 1: Summary of Existing Surveys

Paper	Method	Focus	AD-T	Testbed	Dataset	EM	CUT
Cook et al. [11]	Survey	Time-Series Data	●	○	○	○	○
Himeur et al. [12]	Review	Energy Consumption in Buildings	●	○	○	○	○
Erhan et al. [13]	Review	Sensor System	●	○	○	○	○
Moustafa et al. [14]	Survey	Network Anomaly Detection	●	○	●	●	●
Gaddam et al. [15]	Survey	Sensor Faults and Outliers	●	○	○	○	○
da Costa et al. [16]	Survey	Intrusion Detection in IoT	●	○	●	◐	○
Luo et al. [17]	Systematic Survey	Cyber-Physical Systems	●	●	●	●	○
Benkhelifa et al. [18]	Review	Intrusion Detection Systems	●	○	○	○	○
Fahim & Sillitti [19]	Systematic Review	Intelligent Inhabitant Environments	●	○	○	●	○
Taha & Hadi [20]	Review	Categorical Data	●	○	●	○	○
Fernando et al. [21]	Survey	Medical Data	●	○	●	●	○

○ No, it does not discuss this topic, ● Yes, it discusses this topic ◐ partially discusses this topic. "**AD-T**"=Anomaly Detection Techniques. "**EM**"=Evaluation Metrics. "**CUT**"=Commonly Used Tools in the context of testbed and anomaly detection.

2.1 Intrusion Detection Surveys

The survey of Da Costa et al. [16] investigated intrusion detection techniques in computer networks with a focus on IoT. In addition, a summary of datasets was presented along with their size in terms of the number of samples and the content of the dataset. However, Da Costa et al. [16] did not include readings of IoT devices because they focused on network anomaly detection. Benkhelifa et al. [18] presented a critical review that discussed current threats and practices of IoT, different intrusion detection systems, and different detection techniques. They also proposed an IoT-IDS architecture and provided a description of the proposed system. Although they investigated intrusion detection for IoT, they did not consider anomalies that are not generated by cyberattacks.

2.2 Sensor-Focused Surveys

Erhan et al. [13] published a review that discusses anomaly detection techniques for sensor systems ranging from time series and statistical analysis to supervised learning, RL, and Deep Learning (DL). They also investigated the effect of the deployment of sensors in terms of different architectures, such as cloud, edge, and fog. Furthermore, they presented four common sources of anomalies: environment, system, communication, and various attacks. One of the most challenging aspects of sensor systems anomaly detection that Erhan et al. [13] presented is energy consumption, as algorithms deployed at sensors will consume energy to perform computations to detect anomalies. Similarly, Gaddam et al. [15] focused on presenting a survey about detecting sensor faults and outliers. They also presented a classification of sensor failure detection and outlier identification techniques for IoT, along with the benefits and drawbacks of each technique discussed. Even though Erhan et al. [13] and Gaddam et al. [15] investigated anomaly detection techniques for IoT data, they did not discuss the process of developing a testbed and how the anomalies are generated.

2.3 Medical Surveys

Focusing on medical data, Fernando et al. [21] presented a systematic review that investigates anomaly detection algorithms in medical data, including endoscopy images, heart sound anomalies, and epileptic seizures. Moreover, they provided methods, evaluation results, and limitations for MRI-based anomaly detection. This thorough investigation of anomaly detection techniques in medical data was evaluated on different datasets.

2.4 Data Type Driven Surveys

Cook et al. [11] discussed how IoT data differ from other domains' data because the time series and environment must be considered when analyzing and choosing the anomaly detection algorithm. Furthermore, they presented several challenges in IoT anomaly detection, such as contextual information, dimensionality, noise, and stationarity. They also mentioned current anomaly detection methods in univariate and multivariate time series data but did not include evaluation metrics used to evaluate such algorithms. Taha & Hadi [20] published a review that focused on anomaly detection methods in categorical data. Their paper reviewed different anomaly detection algorithms, such as the Bayesian/conditional frequency and Shannon entropy, in detail, along with their complexity and required parameters. Furthermore, they discussed three issues that face categorical data anomaly detection: the computational complexity

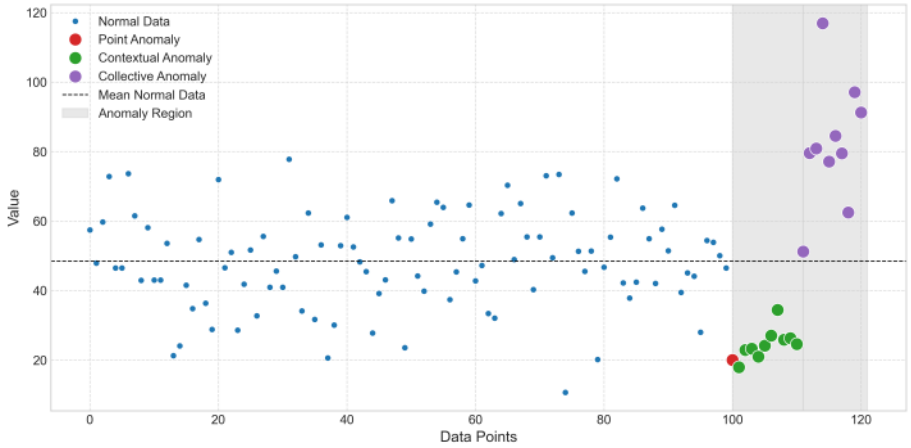

Figure 1: Difference between the three types of anomalies: *point, contextual, and collective* [22][23].

of anomaly detection algorithms, the need for humans to intervene to describe the parameters for anomaly detection algorithms, which is difficult in real-time applications, and the lack of categorical datasets. However, they focused on categorical data only.

2.5 Smart Environment Surveys

Fahim & Sillitti [19] provided a systematic literature review that investigates the detection of anomalous behaviors in intelligent inhabitant environments, intelligent transportation systems, smart objects, healthcare systems, and industrial systems using both statistical and machine learning (ML) methods. Nonetheless, Fahim & Sillitti [19] covered different applications of anomaly detection, but they did not include anomaly detection within a built-in environment. Additionally, Himeur et al. [12] focused on power consumption anomaly detection within buildings. They also provided a taxonomy of the anomaly detection schemes in energy consumption in terms of AI algorithms, application scenarios, detection levels, and computing platforms. They claimed that most anomaly detection work published in this area is detecting anomalies only when there is a surprising increase or decrease in power consumption; however, this is not always the case because a slight difference could be an anomaly. They focused on power consumption, which leaves out a broad range of IoT data.

2.6 Network Surveys

The focus of Moustafa et al. [14] was on network anomaly detection systems. They discussed methods used to detect anomalies in such domain and presented benchmark datasets for evaluating decision engine approaches. Even though they discussed multiple related datasets, they did not discuss how a testbed should be deployed to be able to evaluate the presented anomaly detection techniques.

2.7 Cyber-Physical System Surveys

Luo et al. [17] reviewed state-of-the-art DL anomaly detection methods for cyber-physical systems rather than conventional anomaly detection methods. Additionally, Luo et al. [17] present a taxonomy for types of anomalies, detection techniques, implementation (whether it is a simulation, a testbed, or data from real-world systems), and evaluation metrics. The focus of Luo et al. [17] was on cyber-physical systems.

3 Anomaly Detection

In this section, we highlight different anomaly detection types and techniques that we found in the literature. We also compare different anomaly detection techniques and highlight key points for each presented technique.

Table 2: Comparison Between Different Anomaly Detection Techniques

AD Techniques	Labeled	ML-model Example	Key Points
S	Yes	SGB [24]	Number of anomalies is much less than normal instances
SS	Partially	Ramp-OCSVM [25]	More common than supervised AD due to the fact that they do not need anomalous labeles
US	No	GAN [26]	Produces high false positive if a dataset has more anomalous samples than normal ones.
RL	No	DQN [27]	Needs a thorough analysis of the anomaly and its relationship to the environment.

"**AD**"=Anomaly Detection, "**S**"=Supervised, "**SS**"=Semi-supervised, "**US**"=Unsupervised, "**RL**"=Reinforcement Learning, "**SGB**"=Stochastic Gradient Boosting

3.1 Anomaly Types

According to Chandola et al. [22], there are three main types of anomalies: point, contextual, and collective. A **point anomaly** is the simplest type of anomaly that occurs when a single point is different in relation to other points within the same data set. A **contextual anomaly** occurs when the pattern of the data changes and is considered as an anomaly only in relation to its adjacent data, whereas it would be considered normal if it was taken outside that context of data. Whenever a group of related data shows anomalous behavior in relation to the entire dataset, such an anomaly is referred to as a **collective anomaly**. Even though individual data in a collective anomaly may not be considered an anomaly on their own, their occurrence as a collection is anomalous. To explain and distinguish between the three types of anomalies, let us take weather temperature as an example. If we have temperature readings between 16°C and 22°C for 30 days, but one of the readings is 45°C. In this case, 45°C would be considered a point anomaly. On the other hand, if we have 10 consecutive days of warm weather in a winter month, that would be considered a collective anomaly. Conversely, 45°C might be normal in some places where the weather is hot during summer months but would be abnormal in other places and during other months; this condition would be considered a contextual anomaly. Figure 1 shows the difference between point, contextual, and collective anomalies.

3.2 Anomaly Detection Techniques

Detecting anomalies involves identifying rare events, items, or observations that deviate significantly from standard behaviors or patterns. Anomaly detection techniques fall into three categories: supervised, semi-supervised, and unsupervised. To perform **supervised** anomaly detection technique, data must be labeled as normal and abnormal, a predicting module is trained against a given labeled data, and then the model would classify new data as normal or abnormal based on the trained labeled data. This approach poses two issues. First, there could be far more normal data than anomalous data in the training dataset, which creates an imbalanced classification [22]. Second, having an accurate classification for the dataset is challenging [22]. Using labeled training data and unlabeled data, **semi-supervised** anomaly detection technique builds more accurate models that require less effort to annotate compared to supervised techniques [28]. Additionally, semi-supervised anomaly detection is efficient in cases where abnormal scenarios cannot be easily modeled. An **unsupervised** anomaly detection technique detects anomalies in an unlabeled test set of data solely based on its properties. Furthermore, the unsupervised anomaly detection technique assumes that a dataset has far more normal than abnormal data; otherwise, this approach would produce many false alarms [22]. Each anomaly detection technique identifies an anomaly using either a score or a label. A score is given to each data in each dataset based on the anomaly detection technique, which indicates whether those data are considered an anomaly. As for the label, each data would have a label that has two states, normal and anomalous.

Feature selection is an important aspect in terms of preparing training data for anomaly detection techniques. Feature selection tends to clean the training data from unwanted features, making the predicting model more accurate, less storage requirement, and decreasing computational overhead for the training process [29]. For supervised feature selection, the process objective is to select subset features from a given data so that it could aid in classifying data samples [29]. Unlike supervised feature selection, unsupervised feature selection uses all data as input for the feature selection process, and the chosen learning algorithm is responsible for improving the feature selection with each iteration [29]. Moreover, feature selection methods can be applied to supervised and unsupervised anomaly detection techniques and to semi-supervised anomaly detection techniques [29].

RL is an ML paradigm that was adopted recently in the field of anomaly detection. The basic idea behind RL is that an agent is interacting with an environment and trying to maximize the reward that an environment provides after each action an agent takes, following the trial–and–error approach [30]. An example of applying RL to the field of anomaly detection is the framework presented by Pang et al. [27]. They introduced a framework that uses RL to observe unlabeled and few labeled data for detecting unknown anomalies. Additionally, the framework was evaluated against semi-supervised and unsupervised anomaly detection methods such as DeepSAD [31], DevNet [32], iForest [33], and

REPEN [34]. The previous methods were tested on four popular datasets: NB15 [35], Thyroid [36][37], HAR [38], and Covertype [39].

On the contrary, Müller et al. [40] believe that applying RL to anomaly detection is yet to be considered efficient for several reasons. First, RL is not fully embraced because it is dominated by simplistic and unrealistic evaluation scenarios. Part of this is that anomaly detection in RL is less defined and formalized than in the other domains. Second, most existing works are basically traditional anomaly detection problems presented as RL anomaly detection problems. Furthermore, RL-proposed methods do not tackle RL-specific anomaly detection problems; instead, they reduce them to classic ML tasks in which RL is only the data source. To tackle these issues, Müller et al. [40] proposed several guidelines that would help to implement realistic and practical RL anomaly detection solutions. The first guideline is analyzing anomalies deeply tied to the environment under inspection and exhibiting frequent change instead of simple pattern recognition problems. Another surprising guideline is the absence of reward at the deployment stage, as Müller et al. [40] believe this would help create a real-world scenario. Table 2 shows a comparison between the anomaly detection techniques we discussed.

4 Research Method

The study approach we used follows the guidelines of David Budgen [41] and comprises the following consecutive phases: developing a set of research questions; eliciting keywords from the developed research questions to generate the search queries; selecting the databases where the search must be conducted; specifying filtering criteria such as language and year of publication; skimming through titles and abstracts to eliminate irrelevant and duplicate research papers; reviewing the papers in detail and developing extensive criteria throughout a comprehensive reading; adding different tags to relevant research papers so that it is easier to comprehend what each paper has covered; and assessing the remaining articles using the research questions that were developed at the start.

4.1 Research Questions

This systematic review aims to investigate the different anomaly detection techniques and testbed characteristics in the built environment context. For that reason, the following questions were developed:

RQ 1 What are the different types of anomalies, and how are they generated?

RQ 2 What are the characteristics and properties of a smart home testbed that are useful in the context of developing and testing anomaly detection techniques?

RQ 3 What are the evaluation criteria for anomaly detection techniques within a built environment?

RQ 4 How are smart home and anomaly detection datasets created and analyzed?

4.2 Search Process

This systematic review has searched for related research papers from the following electronic databases shown in Table 3. Additionally, the literature review used three websites that help to find related research papers, as shown in Table 4. For each website, a related research paper (we used [42] as a seed) was fed to the website as a seed; then, each website generated a visual literature map that shows related references to the fed research paper after analyzing it. We chose the aforementioned research paper as a seed because we needed more research papers that discuss implementing a testbed in detail. This approach has introduced strongly related literature that covers the topic of developing and evaluating a testbed. Table 5 shows the list of keywords used during the search process. Additionally, we used the following search string along with the keywords:

```
- smart home OR building and testbed AND anomaly detection
- smart home testbed OR smart building testbed OR testbed
```

4.3 Filtering Criteria and Quality Assessment

After conducting an initial search, we came across 429 research papers that must be filtered. We started reading titles, then abstracts, and conclusions to determine how related a certain paper is according to our literature review topic. Additionally, we created two labels within Google Scholar: one for research papers that discuss testbeds and another for papers that discuss anomaly detection for a built environment.

Table 3: Digital Databases

No.	Digital Database
1	IEEE Xplore
2	ACM Digital Library
3	Science Direct
4	Research Gate
5	Elsevier

Table 4: Websites

No.	Websites
1	Research Rabbit
2	Connected Papers
3	LitMaps

Table 5: Keywords

No.	Keyword
1	smart home testbed
2	smart building testbed
3	testbed
4	smart home anomaly detection
5	smart building anomaly detection
6	testbed anomaly detection

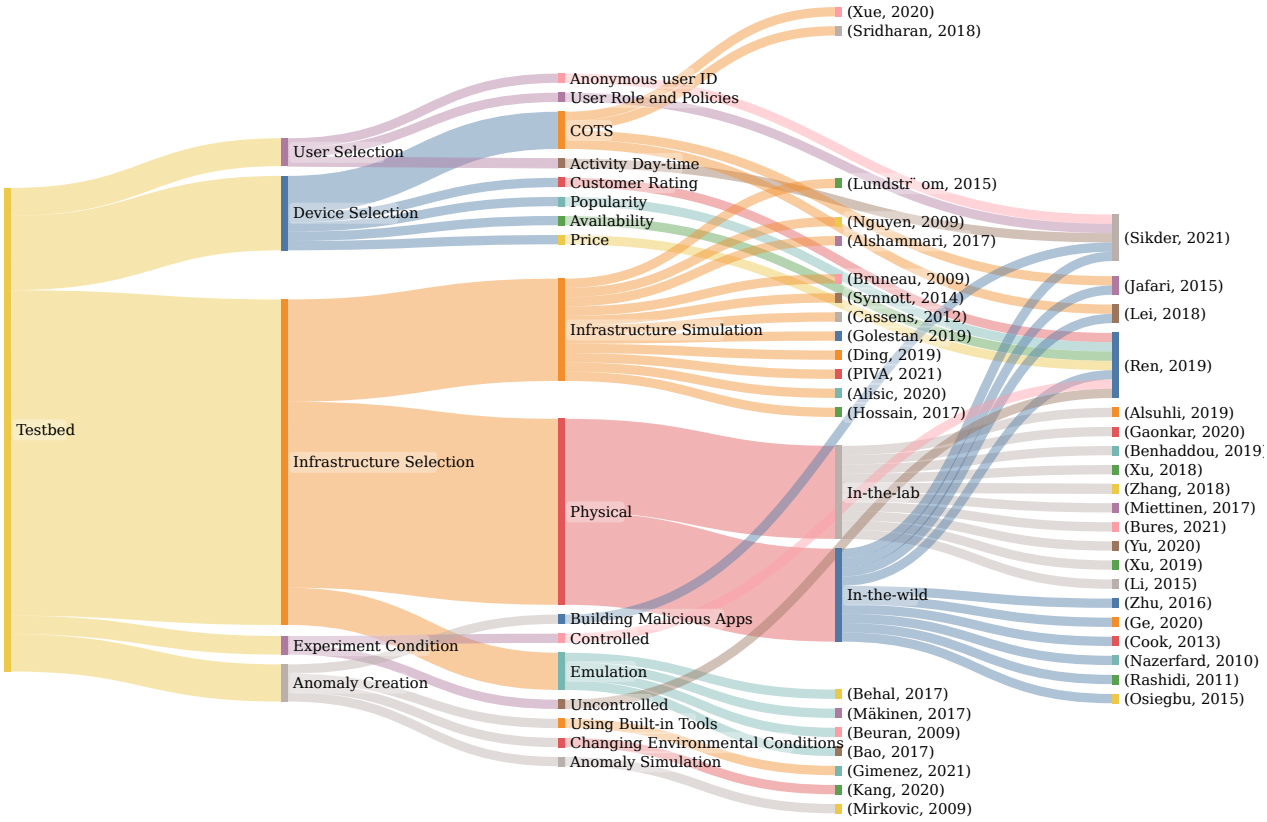

Figure 2: Testbed characteristics in terms of infrastructure, device selection, user selection, experimental conditions, and anomaly creation.

5 Testbeds

Having the testbed as a second-class citizen has always been the case when evaluating anomaly detection techniques. For that reason, we are introducing testbed specifications, infrastructure configurations, user and device selection criteria, and different approaches to creating anomalies as testbed characteristics that we summarize in Figure 2.

5.1 Testbed Design Specifications

Designing and implementing a testbed is not an arbitrary task. There should be guidelines to accomplish the testbed design; as such, we are trying to form testbed development guidelines from the literature review we are conducting. Siboni et al. [43] illustrated several testbed design specifications:

- **Device under test:** The testbed should adopt different types of devices, such as smartwatches and fitness trackers. Even though Siboni et al. [43] were specific about wearable devices, this design requirement should also apply to smart home testbed devices.
- **Testing environments:** The testbed should be able to mimic and simulate different environments, whether static or dynamic, for instance.

- **Security testing:** There should be a variety of security tests conducted on the testbed, each targeting a different aspect of security. A vulnerability scan and penetration test should be conducted to assess and verify the security level of IoT devices.
- **Simulator array:** The testbed should be able to simulate different scenarios to generate real-time data. For instance, this design requirement could be accomplished using a simulation technique like a GPS simulator to try the testbed in different locations. Another simulation scenario is the use of a network simulator to try different wireless technologies such as Zigbee and Bluetooth.
- **Communication channels:** The testbed should support different communication technologies, wired and wireless, such as Zigbee, Wi-Fi, Bluetooth, USB, and Ethernet.
- **Protocol analysis:** The testbed should be able to analyze different types of protocols such as IPv4, TCP, and TLS.
- **Operating system compatibility:** The testbed should provide virtualization so that its devices can run their software, which supports different types of operating systems.
- **Data forensic analysis:** The testbed should maintain stored device data, including data from backup and log files, to conduct data forensic analysis.
- **Management and report mechanisms:** It is important that the testbed is equipped with management and reporting mechanisms in order to be able to control and manage the testing flow. Tools for exploring and analyzing data should be included in these report tools.
- **User intervention and automation:** The testbed should allow for user intervention and automation during all testing phases.
- **Testbed enhancement capability:** The entire testbed should be implemented as a plug-in framework to support future operations.

5.2 Infrastructure Deployment Configurations

This section discusses various testbed infrastructure deployment configurations that include physical (both in-the-wild and in-the-lab), simulation, and emulation. We also summarize the differences between each testbed deployment that Bhatia et al. [44] discussed in Table 6.

5.2.1 Physical:

In-The-Wild:

There are many ways to implement a testbed within the built-in environment in terms of the layout of the environment. Sikder et al. [45] believe that different home layouts directly affect data generated from devices, as different layouts would lead to different user activities. Furthermore, a small home layout might restrict some users' activities. They ran their experiment in different home layouts: single-bedroom apartments, double-bedroom homes, and duplex homes with two, three, and four residents in each apartment or home. Additionally, they used a single smart home system (SHS) where all smart devices are connected to a central hub, a multi-SHS where multiple hubs are used, such as Amazon Alexa and Samsung SmartThings so that different devices are connected to different hubs, yet they all share the same network access point. They found that the accuracy of their framework is better when multiple SHS platforms are used in all different home layouts. Furthermore, a single-bedroom home has better accuracy than other layouts except when adaptive training is used in a duplex home. Having high accuracy when multiple SHS platforms are used shows the relationship between different smart home layouts and the accuracy of the detection algorithm. Following the same layout approach, Lei et al. [46] evaluated their virtual security button (VSButton) prototype, which consists of a Netgear R63000v2 Wi-Fi router, a motions detection module using a Lenovo X200 laptop and an Amazon Echo Dot in three different layouts: a square room, a rectangle room, and a two-bedroom apartment where six participants were involved in their study. Cook et al. [47] also conducted their smart home-in-a-box experiment in a three-bedroom apartment where 20 participants were involved in the study. Each participant was given a smart home-in-a-box kit, and they evaluated how long it takes a participant to install smart home devices.

In-The-Lab/Prototype: Jabbar et al. [48] have taken another approach in terms of testbed infrastructure deployment. They designed a prototype for their smart home called IoT@Home, which consists of a master bedroom, two bedrooms in addition to the master one, two toilets, a kitchen, a living room, and a porch using software called NX10. The smart home's details include sliding motions used to close and open the windows. A steel rod serves as the base for the upper part of the windows. A motor is installed on the upper side of the doors to open and close the doors. A motion sensor attached to the door provides automatic closing and opening functions. To enter the house, the owner scans the

Table 6: Infrastructure Deployment Comparison

Characteristics	Physical	Simulation	Emulation
Realism	Optimal realism	Lack of Realism as it relies on software	More realistic as it uses real machines that run an OS, while part of the testbed is virtualized such as routers
Easy to Reconfigure	No	Yes	Moderate as part of the testbed depends on physical machines
Cost	High	Low	Moderate as part of the testbed is virtualized
Scalable	No	Yes	No
Examples	CASAS, Aegis+	OpNet, NS-2	Emulab, DETER

access card with an RFID sensor located outside the house. To make water flow through the house, a water tank is also installed in a high position outside so water can flow smoothly. An ultrasonic sensor is installed in the tank to detect the level of the water in the tank. Additionally, the house circumference is 100 cm by 100 cm. In conclusion, smart home layouts can differ in terms of their implementation, where researchers have used a real physical layout or built a prototype smart home.

5.2.2 Simulation:

A smart home environment could also be simulated by monitoring smart home devices and then building a model that simulates each device's behavior. Pina et al. [49] followed this approach where they decided to use a temperature sensor, presence sensor, light bulb, simple TV, joystick, and remote control that would be installed in different rooms from E1 to E4 that have different layouts. Afterward, they used the Markov chain to simulate each device's behavior after monitoring its state and how it changes and communicates with other devices. Even though this approach might be suitable in cases where budget is an issue, there is no guarantee that the simulated states will truly reflect how devices communicate and behave in the real world.

5.2.3 Emulation:

To test their DDoS detection mechanism, Behal et al. [50] used the Common Open Research Emulator (CORE) to increase the number of nodes in their testbed. In their emulated testbed, they deployed 16 virtual nodes and 4 soft routers using CORE, and they used 75 physical nodes in three different groups of 25 computers each, 3 D-Link physical routers, 4 L2 switches, 2 L3 switches, and 1 2-processor 8-core Linux server that acts as the victim Web server. Although this approach used software, CORE in this case, and hardware to build the testbed, the result of using such an approach is not widely adopted in the testbed community.

5.3 Selection Criteria

As we discuss testbed requirements and characteristics in the context of a built-in environment, we must also discuss users using the testbed, experiment conditions, and device selection.

5.3.1 User Criteria

In this section, we discuss user selection criteria in the context of the smart home environment so that the smart home testbed can generate realistic data that reflects the user's behavior. According to Sikder et al. [45], the following selection criteria should be considered:

- **Anonymous user ID:** Each residence or user should be assigned an ID, and this ID should be used for all communication with the user to maintain the user's privacy.
- **User role and policies:** Each residence can choose a rule to their liking. For example, a user can choose to turn on the light once a motion sensor detects a motion. Conversely, another user can choose to turn the light differently once the door is open, for example. Different rules can be assigned by different users.
- **Activity day-time:** Different time activities should be considered because an individual's activity differs in time. A working adult might have more activity on weekends than on weekdays. Hence, a change in daily routine might not be malicious behavior.

5.3.2 Device Criteria

One of the tasks to develop a testbed is to select devices for the testbed. As several devices come from the same category, there should be a way to prefer one over the other. During the selection of IoT devices, Ren et al. [51] selected IoT devices that fall into a broad range of categories based on popularity, customer rating, price, and availability for both the UK and the US because the study wanted to compare between the two testbeds in terms of IoT device information exposure in different locations.

5.3.3 Experiment Condition

Smart home experiments can take place in a controlled and uncontrolled environment. Ren et al. [51] conducted an experiment on how much IoT device information is exposed by creating a testbed in a studio apartment. In their study, 36 participants were involved over the course of 6 months, so the testbed was examined in an uncontrolled environment in the US only and in a controlled environment both in the US and the UK. In an uncontrolled setting, participants were allowed to enter the lab and use the devices at any time of the day except when another experiment condition was taking place. Conversely, controlled environments are restricted to certain times. One privacy violation discovered in this study is that the door bill was recording and sending videos to the service provider without the consent of recorded individuals. The exposed data were categorized into stored, sensor, and activity. Furthermore, Ren et al. [51] categorized parties involved in IoT device shared data: first party, which are vendors; support party, which are companies who offer services such as cloud services; and third party, which are analytical or advertising companies.

5.4 Anomaly Creation

This section discusses several techniques to create different types of anomalies within the smart home environment.

5.4.1 Building Malicious Apps:

One of the anomaly-creation methods is building a malicious app that mimics a cyberattack to evaluate the anomaly detection algorithm proposed. Sikder et al. [45] followed this approach, creating threat scenarios at first for each attack they would like to perform. In this article, five different threat attack scenarios were considered (impersonation attack, false data injection, side channel attack, denial-of-service, and triggering a malicious app). For **impersonation attack**, an app was built to leak the smart unlock code via SMS to the attacker so the attacker could impersonate the owner of the smart home in this case. Another malicious app for impersonation attacks is where an unauthorized smart home user replays a voice command captured from a valid user, impersonating a valid user. For **false data injection**, an app was built to inject false data into sensors, and a voice command was injected into smart home devices. For **side channel attack**, two malicious apps were built: one flickers light in a pattern while there is no user in the room, and the other is a malicious app that sends a signal that would cause a smart speaker to perform an action that is legit but harmful. For **denial-of-service**, an app was built to disable any running smart home task. For **triggering a malicious app**, an app was built to change the state of a device that would lead to a malicious act on a smart device.

5.4.2 Using Built-in Tools:

Creating cyber anomalies within a testbed is crucial. One must be positive that any cyber anomalies within the testbed do not spread to other networks or unintended devices. Several tools have been presented to accomplish such a task. Gimenez et al. [52] generated several wireless technology attacks that were built using **Mirage** (github.com/conchyliculture/mirag-e)) open-source framework. For Zigbee technology, Zigbee scan and Zigbee injection classes of anomalies. For Wi-Fi technology, de-authentication and rogue AP classes of anomalies were generated. For Bluetooth Low Energy (BLE), a BLE scan class of anomalies was generated. Furthermore, each of the previous anomalies was conducted for 2 minutes multiple times at different locations.

Dadkhah et al. [53] also presented **Low Orbit Ion Cannon (LOIC)** (`github.com/NewEraCra-cker/LOIC`)), which they used to generate DDoS attacks using HTTP, UDP, and TCP protocols. Real-Time Streaming Protocol (RTSP) brute force attacks were performed using **Nmap** (nmap.org) scanner tool and **Hydra** (github.com/vanhauser-thc/thc-hydra) to find URLs for RTSP cameras and watch the camera stream.

5.4.3 Changing Environmental Conditions:

Another method of creating an anomaly within a built environment is injecting anomalies by changing a factor within the physical environment, such as temperature. Kang et al. [54] created a testbed using Mi Air Purifier 2S and Mi Sterilization Humidifier to run an anomaly detection experiment. Mi Air Purifier 2S was turned on in their

Table 7: Anomaly Detection Mechanism

Paper	Anomaly Detection Algorithm	Testbed	Dataset	CA	PA	DF
Sikder et al. [45]	Markov Chain–based machine learning model	●	○	○	●	●
Ullah & Mahmoud [59]	FFN	○	●	●	○	○
Kang et al. [54]	(Not mentioned in the paper)	●	○	○	●	●
Hosseini et al. [60]	Gaussian-based Kernel Density Estimation	●	○	○	●	●
Li et al. [61]	LSTM Autoencoder	○	●	●	○	○
Liu et al. [62]	AMCNN-LSTM (Attention Mechanism-Based CNN-LSTM Model)	○	●	●	○	○
Mudgerikar et al. [63]	Random Forest	○	●	●	○	○
Djenouri et al. [64]	The genetic algorithm and the bee swarm optimization	○	●	●	○	○
Mothukuri et al. [65]	LSTM & GRU	○	●	●	●	●
Shafi et al. [66]	E3ML & MLP &RNN &ADT	○	●	●	○	○
Yahyaoui et al. [67]	OCSVM & DL	●	●	●	●	○
Gimenez et al. [52]	Autoencoder neural network & temporal and spatial diagnosis	●	○	●	○	●
Heartfield et al. [4]	Isolation Forest	●	○	●	○	○
Teh et al. [68]	RFE & RFR for feature extraction and selection &PCA for AD	○	●	○	●	○
Sater & Hamza [69]	Three LSTM layers	○	●	●	○	○

○ No, it does not discuss this topic, ● Yes, it does discuss this topic. "**CA**"=Cyber Anomaly. "**PA**"=Physical Anomaly. "**DF**"=Device Fault.

experiment, and Mi Sterilization Humidifier generated air mist as an environmental condition change (creating a polluted environment) to inject anomalous values into the air purifier.

5.4.4 Simulation:

Even though one could argue that simulated anomalies lack realism, they are viable options in case creating actual anomalies is difficult for budget reasoning. Said et al. [55] implemented an intrusion detection model in a smart hospital IoT system to identify events that matter about patient health, environments, and cyber threats. Using Contiki Cooja's (`github.com/contiki-ng`) simulator, they implemented and tested the model. They used multiple simulation parameters such as the number of nodes, topology, area, sending rate, simulation run time, and physical layer. The proposed approach detects 93.4% of rank attacks and 60.8% of flood attacks, which is not a high detection rate for flood attacks.

Similarly, Mirkovic et al. [56] implemented a DoS attack using a network simulator. In their paper, they compared the use of a simulator and testbed settings in evaluating attack metrics by calculating acceptable, scientifically determined thresholds for several QoS requirements. NS-2 was used to create a simple network, a client, a server, and an attacker. Then, they generated several network traffic and DoS attacks on the simulator. They also ran a DoS attack on the DETER testbed [57], and the result was that the testbed could handle much higher throughput than the NS-2 simulator due to the failure of some transactions in the simulator.

Lundström et al. [58] presented an anomaly detection mechanism for human activity in smart homes that is trained on simulated normal data using the Markov model-based approach. The model was created using prior knowledge from data collected from different smart homes. Different abnormal data were also simulated to be tested on the proposed approach. According to Lundström et al. [58], the proposed approach achieved a detection accuracy of 85% in detecting anomalies.

6 Anomaly Detection Mechanism

In this section, we discuss several anomaly detection mechanisms that were applied on a testbed and on different datasets in terms of anomaly detection algorithm used, dataset choices, devices used in testbeds, anomaly type, and evaluation metrics. A summary of anomaly detection mechanisms is presented in Table 7

6.1 Testbed-based Anomaly Detection:

Sikder et al. [45] presented AEGIES+, a security context-aware framework that monitors different smart devices and user activities to detect any abnormal behavior. The proposed framework took into consideration different features of the smart home environment. The framework demonstrates how sensor devices within smart homes depend on each other, even if it is not that obvious. For example, a smart light bulb could be configured only to be turned on once

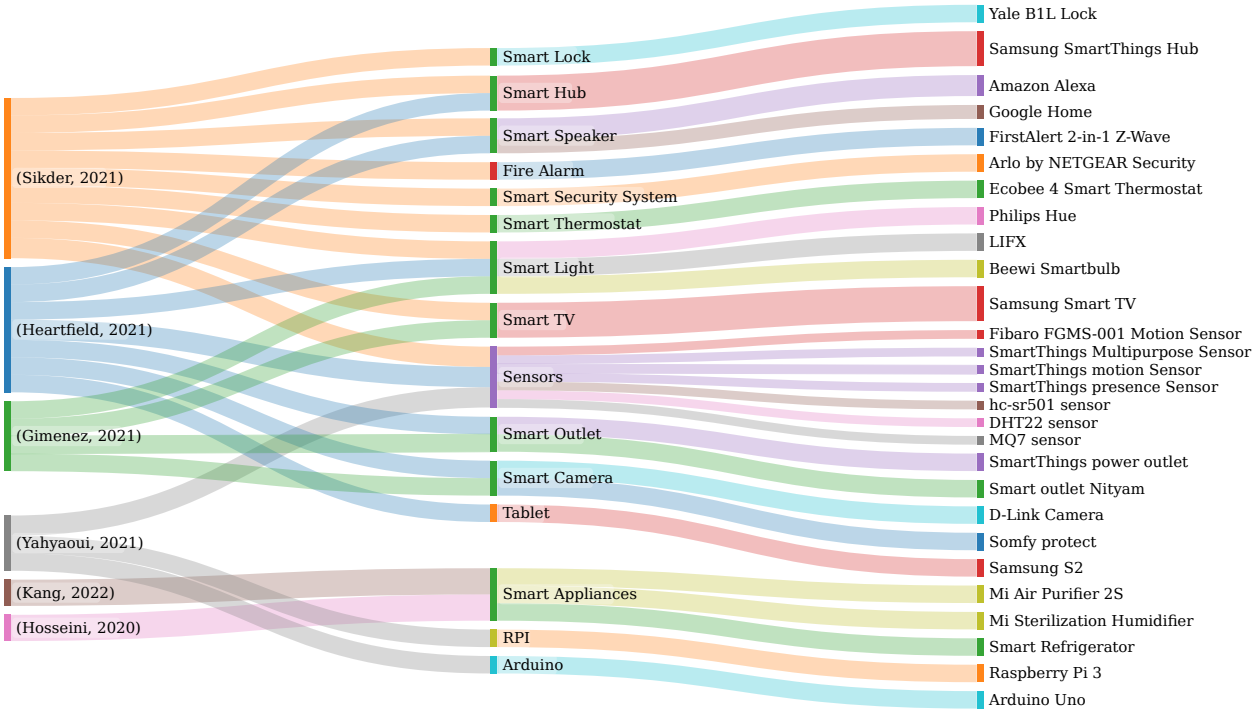

Figure 3: Testbed devices that were used in our primary selected studies where we could see common IoT devices and sensors such as SmartThings contact sensor. These devices and sensors are from real-world testbeds, not from datasets.

someone is in the room that the light bulb is in. One way to do that is to have a motion sensor installed, and once it detects a motion, it notifies the smart light bulb that there is someone in the room so it can be turned on. This shows how sensors in the smart home environment depend on each other. Furthermore, users interact with sensors differently, depicting whether a certain action is malicious. This is achieved by monitoring usual user activity and determining whether a certain action is a common user behavior.

Sikder et al. [45] developed a testbed consisting of a Samsung SmartThings hub, Amazon Alexa, Google Home, Philips Hue light bulb, Yale B1L lock with Z-Wave Push Button Deadbolt, FirstAlert2in-1Z-Wave Smoke Detector, and Carbon Monoxide Alarm, Arlo by Netgear Security System, Ecobee 4 Smart Thermostat, Samsung6Series UN49MU6290F LED Smart TV, Fibaro FGMS-001 Motion Sensor, and Samsung Multipurpose Sensor. The testbed was implemented in single-bedroom apartments, double-bedroom homes, and duplex homes with two, three, and four residents in each apartment or home. Furthermore, Sikder et al. [45] took into consideration different implementation goals that were presented in 5.3.1 along with the home layout where the most common house layout in the US was chosen to emulate real-life scenarios. They also considered user selection criteria as discussed in 5.3.1.

To test the AEGIES+, five different threat attack scenarios were created and presented in 5.4.1 using malicious apps for each attack scenario, hence creating anomalies. Thereafter, True Positive (TP), True Negative (TN), False Positive (FP), False Negative (FN), Accuracy, and F1-score metrics were used to calculate recall and evaluate the proposed framework.

Yahyaoui et al. [67] presented a Reliable Event and Anomaly Detection Framework for the IoT (READ-IoT), which detects abnormal events and anomalies where they mimic a fire incident and unauthorized access for a person. Using Wing Python IDE, they developed event and anomaly detectors, then deployed them using docker containers in fog and cloud environments. The ML components were developed using TensorFlow and scikit-learn libraries in Python. Each cluster in the prototype consists of a cluster head and three sensors (Arduino Uno nodes): two motion detection nodes based on PIR sensors (HCR501), one fire detection node based on temperature-humidity (DHT22 sensor), and a gas sensor (MQ7 sensor). Sensor nodes communicate with their respective Raspberry Pi 3 nodes using Zigbee protocol, which supports indoor and outdoor communication with a range of 75–100 and 300 m, respectively. An encryption algorithm of 128 bits and a message integrity code were used to secure the communication. To test their anomaly detection system (ADS), they used a network simulator called Castalia 3.2 (omnetpp.org/download-items/Castalia.htm) to simulate normal traffic and inject malicious traffic. They used a simulator to test their approach and a KDD dataset. Different deployment scenarios were considered for testing their event detection system, where the entire structure

is implemented in the cloud, fog, hybrid, and READ-IoT. The result shows that READ-IoT has the least end-to-end delay in detecting fire and unauthorized person events, thus making it the best performance approach. As for the ADS, the proposed approach detected malicious traffic with an accuracy of over 90% and 98% using one-class support vector machines (OCSVM) and DL, respectively. Yahyaoui et al. [67] also noted that the use of DL for training is time-consuming and needs high bandwidth.

Kang et al. [70] built a testbed consisting of Mi Air Purifier 2S and Mi Sterilization Humidifier to test their anomaly detection method for smart home devices. They divide anomalies into four main categories: abrupt, accumulated, functional component, and human. An abrupt anomaly happens when a sudden change occurs, and an accumulated anomaly happens when the data are within the normal range, but the overall readings of the data are not the same as the usual readings of such a device. A functional component anomaly happens when a device is faulty and does not perform its functions properly. Obviously, the human anomaly is an anomaly that is caused by humans. For generating anomalies, Kang et al. [70] artificially injected anomalies into the air purifier in real time and monitored how the threshold value changed according to the generated anomaly. Even though they evaluated the proposed anomaly detection solution for smart homes, only two smart devices were used in their study.

The smart refrigerator was also tested for anomalies as presented by Hosseini et al. [60]. with a focus on the power consumption of the smart refrigerator. Anomalies were divided into faults and abnormal behaviors. Two scenarios were considered for faults: the door is not closed, and the door rubber seal is defective. To mimic those scenarios, the door was left open during different periods, and the rubber seal was bent out of shape. Two scenarios were considered for abnormal behaviors: opening and closing the refrigerator and having a full refrigerator. To mimic those scenarios, the door was opened and closed many times of the day, and the refrigerator was filled with water at different temperatures with different amounts, respectively. Hosseini et al. [60] used a Gaussian-based kernel density estimation as an anomaly detection algorithm and were able to detect anomalies with 99% and 98% accuracy when smart and standard refrigerators were used, respectively. A summary of testbed devices is presented in Figure 3.

6.2 Dataset-based Anomaly Detection

An edge-assisted anomaly detection framework called ADRIoT was introduced by Li et al. [61] According to the proposed framework, the anomaly detection module has three layers: *capturer*, *pre-processor*, and *detector*. The **capturer** layer simply captures incoming network packets. In the **pre-processor** layer, network traffic is vectorized and fed to the ML algorithm. It also hides network characteristics by converting consecutive incoming network packets into a sequential stream of data. The **detector** layer is where the proposed framework detects an anomaly. In this layer, the raw network bytes are used as a feature for the ML algorithm. The framework uses long short-term memory (LSTM) autoencoder and is evaluated using TP, TF, FP, FN, receiver operating characteristics (ROC), area under the curve (AUC), and equal error rate (EER) evaluation metrics. The proposed framework was also evaluated with different unsupervised anomaly detection algorithms: OCSVM, isolation forest, and robust covariance.

Furthermore, Li et al. [61] tested the proposed framework on the IoT traffic dataset that was published by Northeastern University and Imperial College London [51]. The dataset was collected from two smart homes; one is in the US, and the other is in the UK. This location diversity is beneficial because it illustrates how an IoT device would behave in different locations. The dataset gathered traffic from the following IoT devices, where the same devices were used in US and UK locations, along with the size of each pcap file for US and UK locations consecutively: Apple TV (44.8 and 66.9 MB), Nest thermostat (16.2 and 33.3 MB), Wemo Wi-Fi plug (55.7 and 47.8 MB), TP-Link Wi-Fi bulb (36.4 and 12.6 MB), TP-Link Wi-Fi plug (15.1 and 7.4 MB), Zmodo doorbell smart camera (147.9 MB and N/A), Echo Plus speaker (56.5 and 35.6 MB), Roku smart TV (42.4 and 26.6 MB), Sengled IoT Hub (3.2 and 1.6 MB), and Echo Spot speaker (45.6 and 87.6 MB). The dataset contains not only IoT traffic but also network payload.

Mudgerikar et al. [71] also presented an edge-assisted intrusion detection system called E-Spion. The anomaly detection engine is divided into three distinct detection modules: *Process White Listing Module (PWM)*, *Process Behavior Module (PBM)*, and *System-call Behavior Module (SBM)*. **PWM** is the least expensive and uses a whitelisting-based method. Their technology analyzes all benign processes operating on the device during the learning period and creates a device-specific whitelist of all running processes. During the operation stage and in order to identify anomalous new processes, the system collects and compares the names and PIDs of all running processes with the device whitelist. **PBM** keeps track of the behavior of all running processes on the device and alerts you if one of them is acting strangely. During the learning stage, PBM keeps track of several parameters for each running process on the device. After gathering logs during the learning stage, they trained an ML classifier for each device to detect aberrant process behavior. Eight metrics or features were extracted from the parameters and used in the PBM classification model. PBM is more expensive than PWM, but it gives finer detection and can identify more complex malware that can masquerade as benign processes instead of generating new dangerous processes. **SBM** keeps track of how each process on the device behaves in response to the system calls it makes. On the one hand, SBM is the most expensive module, but on the other hand, it

offers the most precise and accurate detection approach. SBM monitors 34 different types of system calls issued by each active process. Mudgerikar et al. [71] track four metrics or features for each type of system call recorded: the number of calls made (1), percentage of time consumed (2), total time taken (3), and average time taken per call (4).

Each network device must undergo four stages: initialization, learning, operation, and anomaly detection. For the initialization on the edge server, a new empty device profile and public–private key pair are established when a new device joins the network. Then, the private key is uploaded to the device, and this key pair is used to secure all sessions between the edge and the device. For learning, the edge server obtains the device's PWM, PBM, and SBM records and creates a single three-layered baseline profile for it. The training dataset for these modules is made up of data acquired from the PBM and PWM logs during the learning stage.

To evaluate their approach, Mudgerikar et al. [71] downloaded malware samples from IoTPOT [72], VirusTotal, and Open Malware datasets. For anomaly detection, they tried different binary classifiers: naive Bayes, decision trees, logistic regression, random forest (RF), K-nearest neighbor, and multilayer perception. The result was that the RF classifier was the best in terms of detection intrusion, and tree-based classifiers were the fastest to build.

Ullah & Mahmoud [59] presented an anomaly detection framework in IoT networks using a generative adversarial network (GAN). They used different types of GAN to generate synthetic data to balance the datasets on which they wanted to test the framework and then applied the detector model. They tested the proposed framework on the following datasets: KDD [73], NSLKDD [74], Bot-IoT [75], IoT-23 [76], IoT Network Intrusion [77], MQTT-IoT-IDS2020 [78], MQTT-set [79], which had an average detection rate of 99.05%, 98.85%, 97.70%, 97.30%, 94.52%, 98.66% and 99.10%, respectively, when using binary class conditional GAN as a generative model and Feed Forward Neural Network as a detector model.

Mothukuri et al. [65] proposed a federated learning (FL) anomaly detection approach that detects intrusion in IoT networks where LSTM and gated recurrent unit neural network models can be trained on IoT networks without transferring network data to a centralized server. The architecture of their proposed approach consists of a **virtual instance** that was created using PySyft where they created a number of end devices and a central FL as a server so that the ML parameters can be shared between end devices and the central FL server, **preprocess of captured data** where pcap files are generated from network traffic and converted to CSV files to clean up unwanted features, **FL training** where each client node executes training rounds with their data and shares the weights of their local ML models to the central model, and an **ensembler** that enhances the accuracy of ML models by combining their outputs. The proposed approach was evaluated on a Modbus-based dataset [80], which has the following attacks: man-in-the-middle, ping DDoS flood, Modbus query flood, and SYN DDoS. Based on the evaluation metrics, they propose a version with FL that achieves greater accuracy with fewer epochs than the non-FL version, where the average accuracy for the FL and non-FL approach is 90.255%, whereas it is 86.134%, respectively. Figure 4 shows datasets and research paper that we discussed in 6.2.

7 Datasets

Having IoT devices in a controlled environment facilitates monitoring and observing how devices communicate and behave under different test scenarios. In this section, we discuss different infrastructure setups, data creation, data collection, capture duration, and methods of dataset analysis.

7.1 Infrastructure Setup

When designing an infrastructure for a dataset, one must consider if the dataset infrastructure will be implemented physically or virtually. For example, Vigoya et al. [81] implemented a virtual infrastructure by building four client nodes, each having four processes where each process represents a temperature sensor. Each sensor (process) has an MQTT client, which sends its temperature data to a central MQTT broker using the MQTT publisher–subscriber approach. Similarly, Koroniotis et al. [75] also implemented a virtual infrastructure that follows the MQTT publisher–subscriber approach, which is implemented over TCP/IP using Node-Red to simulate IoT devices.

On the contrary, Dadkhah et al. [53] followed another approach in terms of infrastructure setup for their dataset where they built a physical infrastructure using two network interface cards: one is connected to the gateway while the other is connected to a NETGEAR GS308 switch, which all IoT devices that require Ethernet are connected to it. Vera Plus smart hub is connected to the switch, acting as an Internet gateway for Wi-Fi IoT devices. Furthermore, Philips Hue Bridge, SmartThings Hub, and Fibaro Home Center Lite have been used to facilitate the communication of Zigbee and Z-Wave devices and sensors. These two examples of infrastructure show how different dataset creation can be in terms of infrastructure. Additionally, it illustrates the possibility of creating a data structure with limited resources.

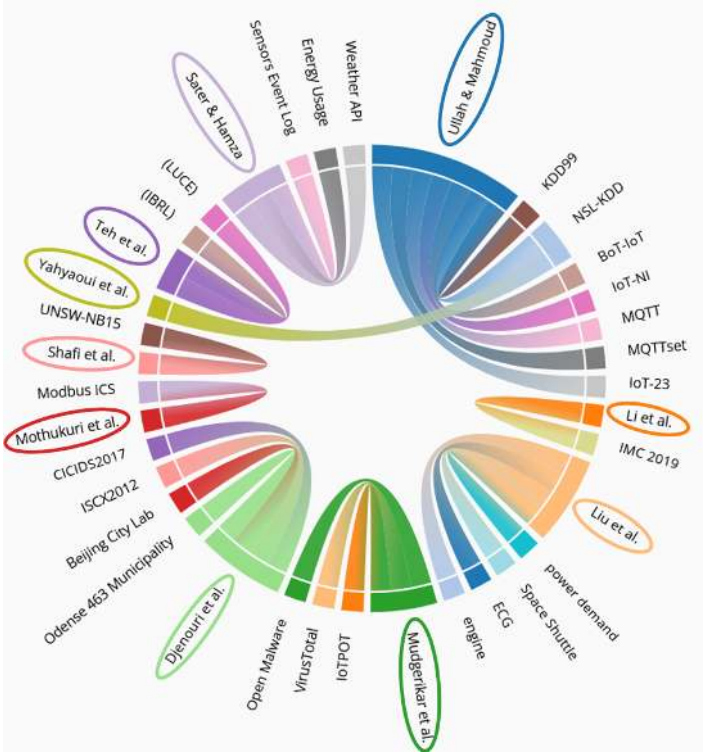

Figure 4: This figure shows datasets that are used by our primary research papers. We highlight each paper with an oval around the authors' names. In addition, the figure shows the link between each research paper and the datasets it uses.

7.2 Network Configuration

Different IoT device testing dictates different network configurations. Some devices only communicate through Zigbee protocol, and others communicate through Wi-Fi and voice commands. This heterogeneity directly contributes to the choice of network configurations. Even though having different wireless communication technologies is beneficial in terms of including a different range of IoT devices, it mostly comes with the burden of having a hub for each wireless communication technology, such as Zigbee. Moreover, some devices within the same wireless communication technology require different hubs, as does Bosch Twinguard, which requires a Bosch smart home controller. Therefore, types of devices and their communication settings must be considered when building the network configuration of a dataset environment.

7.3 Data Collection

The goal of the dataset has an important role in selecting what type of IoT data to collect, as some datasets are only interested in cyberattacks, and others extend their interests to different types of generated data. According to Dadkhah et al. [53], ideally and to observe how IoT devices behave under different scenarios, six statuses must be taken into consideration:

Power: During the power state, all devices must be unplugged from the network, then the network is rebooted. After that, each device is connected to the network on an individual basis, and the MAC address associated with each device is noted, and the network traffic for each device is captured individually as well. Moreover, the capture process should continue until no remaining packet from that device is received. This process would provide an idea of how each device behaves individually.

Idle: During the idle state, an assumption that devices will not communicate with each other during the idle stage must be established. As a result, this stage should usually take place during the night or during holidays. This process would help to identify if a device is making unwanted communication. Ideally, the data captured in the idle stage should match the data captured in the power state for each device.

Interactions: During the interactive state, each device's communication with other devices must be captured. It is also important to mention that some devices have more than one way of communication. For instance, a Philips Hue smart light can be turned on and off through voice command using a voice assistant device (such as Echo Dot) and the Philips Hue app. Thus, different control conditions must be considered as well.

Scenarios: During this state, the scenario intended for the dataset is performed, and network traffic is captured.

Active: During this stage. all IoT device network traffic is captured, including communications between devices, when there are people interacting with the devices either actively or passively.

Attacks: During attacks, a series of cyberattacks are performed on the devices, and the network is captured during attacks. Several tools can be used to generate cyberattacks, which we discussed in 5.4.2 Section.

Network traffic capture can be achieved using one of the network sniffer tools such as Wireshark and Tcpdump. However, some protocols might require additional hardware or software to be captured, such as the Zigbee protocol.

7.4 Data Creation

Data creation must start once the dataset environment is set up and configured with the required tools. During data creation, several scenarios are generated and followed to achieve the dataset's goal, whether for a smart home dataset or otherwise. For instance, Dadkhah et al. [53] generated six different scenarios that cover several activities.

Coming home: The initial setup for this scenario is that the Ring alarm is armed, and devices are off because no one is home. Then, once a participant enters the lab, the Ring alarm, the Netatmo camera, and ArloQ motion detection are alerted. The participant enters the passcode to disarm Ring and voice commands Google Nest Mini to turn on the Philips Hue lights and Globe lamp. Similarly, the participant voice-commands Alexa to turn on the Atomi Coffee Maker and to play music over the Sonos speaker. Finally, the participant starts the Roomba vacuum cleaner using the associated mobile app to start the cleaning process.

Leaving home: The initial setup for this scenario is that all devices are turned on to simulate the presence of people in the home. Then, the participant voice-commands Alexa to turn off the Atomi Coffee Maker and stop the music that was playing over the Sonos speaker. Similarly, voice commands Google Nest Mini to turn off the Philips Hue lights and Globe lamp. Finally, the Roomba vacuum cleaner to its home base to stop the cleaning process.

Home intrusion (during the day): The participant enters the lab; Ring alert, Netatmo camera, ArloQ, and HeimVision camera motion detection are triggered. The D-Link Water Sensor is also triggered to indicate that the intruder has spilled some sort of liquid. Once the Ring alarm goes off, the participant leaves the lab, which will also trigger Netatmo and ArloQ motion detection. This attack scenario occurs with lights on.

Home intrusion (during the night): The participant enters the lab; Ring alert and Netatmo camera are triggered. ArloQ and HeimVision camera sound detection are triggered. The D-Link Water Sensor is also triggered to indicate that the intruder has spilled some sort of liquid. Once the Ring alarm goes off, the participant leaves the lab, which will also trigger Netatmo and ArloQ motion detection. This attack scenario occurs with lights off to evaluate the night vision capability of the Netatmo camera and the sound detection capability of ArloQ and HeimVision because they do not support night vision.

Owner error: This scenario is to simulate homeowner errors like entering the wrong password for a security lock. The participant enters the lab, which triggers a Netatmo camera and the ArloQ motion detection. Then, the participant enters an incorrect password for the Ring alarm thrice every 30 seconds, setting the alarm off.

IoT vs Non-IoT: This scenario is to test how well IoT devices integrate with Non-IoT devices. To do so, a participant asks Alexa to play music over a Sonos speaker, let LG TV play a YouTube video, surf social media using a phone, browse the Amazon website from a computer, and ask Google to turn the Philips Hue lights on.

Another approach to generating synthetic data is using simulation. Moustafa and Slay [82] used a network traffic simulator called IXIA PerfectStorm, which simulates different types of cyberattacks. Furthermore, Vigoya et al. [81] discussed three ways of generating anomalous traffic: interception, which could be achieved by deleting sent traffic packets; modification, which could be achieved by changing sensor readings that are sent randomly (for instance, Vigoya et al. [77] have changed temperature samples sent to the MQTT broker); and duplication, which could be achieved by sending more data than planned in the first place (for instance, Vigoya et al. [77] sent more tokens than planned).

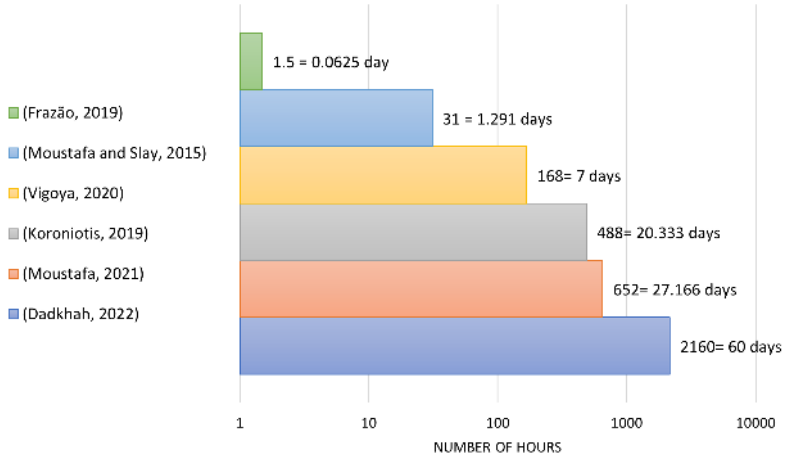

Figure 5: Data capture duration for 6 different datasets.

Table 8: Formulas of Statistical-based Methods

Statistical-based Methods	Formula	No.
Low variance	$p(1-p)$	(1)
T-score	$\lvert\mu_1 - \mu_2\rvert / \sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}$	(2)
Chi-square score	$\sum_{j=1}^{r}\sum_{s=1}^{c} \frac{(n_{js}-\mu_{js})^2}{\mu_{js}}$	(3)
Gini index	$\min_{\mathcal{W}}\left(p(\mathcal{W})(1-\sum_{s=1}^{c} p(C_s\lvert\mathcal{W})^2) + p(\overline{\mathcal{W}})(1-\sum_{s=1}^{c} p(C_s\lvert\overline{\mathcal{W}})^2)\right)$	(4)
CFS	$CFS\_score(\mathcal{S}) = \frac{k\overline{r_{cf}}}{\sqrt{k+k(k-1)\overline{r_{ff}}}}$	(5)

7.5 Data Capture Duration

Data capture duration differs from one dataset to another. In this section, we try to estimate the minimum requirement for dataset capture duration depending on well-known different datasets that were published. (Dadkhah, 2022) [53] stated that the experiments conducted on the IoT devices took 3 months to create the CIC IoT 2022 dataset (www.unb.ca/cic/datasets/iotdatas-et-2022.html). He specifically stated that he captured 30 days of idle network traffic and 30 days of active network traffic (see section 7.3 for more illustrations about different statuses). Furthermore, (Moustafa, 2021) [83] created an IoT dataset called TON IoT (research.unsw.edu.au/projects/toniot-datasets) that has a data capture of approximately 27 days. Even though (Moustafa, 2021) [83] did not specify the exact capture duration, we were able to estimate a data capture duration by calculating the difference between the first and second capture timestamps. Similarly, (Koroniotis, 2019) [75] published Bot-IoT (research.unsw.edu.au/projects/bot-iot-dataset) dataset that has a capture duration of approximately 20 days, where we investigated the first and last capture timestamps to estimate the capture duration.

(Vigoya, 2020) [81] created a dataset (DAD) (github.com/dad-repository/dad) that captured network traffic for 7 days. During those 7 days, different types of anomalies were generated within 5 days, which leaves 2 days with normal network activity. Additionally, each day of the capture duration has a summary of the attack performed, if any; otherwise, it will state that no attack has taken place.

On a shorter duration period, (Moustafa and Slay, 2015) [82] captured traffic for 2 days, where they simulated traffic, using IXIA PerfectStorm, for 16 hours on the first day and 15 hours on the second day, which generated 100 GB of traffic in total for their UNSW-NB15 dataset (research.unsw.edu.au/projects/unsw-nb15-dataset). (Frazão, 2019) [80] also captured network traffic for a short period. His experiment captured network traffic for 1 hour and 30 minutes with 1, 5, and 15 minutes of attack times. The reason behind the varying capture duration is that he wanted to see how each ML classifier would behave with different capture duration. Figure 5 presents a summary of capture duration.

7.6 Data Analysis

After creating the dataset, the data captured must be analyzed so that the dataset can be used in different applications that are tailored to user needs. In this section, we discuss two techniques of dataset analysis.

7.6.1 Feature Analysis

Analyzing network features helps in giving insight into the traffic pattern and whether the traffic is malicious or not. Tavallaee et al. [84] analyzed the well-known KDD CUP 99 dataset features into three categories:

Basic feature: This category has all fields that are extracted from the TCP/IP connection, which has information such as IP and MAC addresses.

Traffic feature: A feature computed on the basis of window intervals is included in this category, which can be further divided into two subcategories: same-host features, which measure protocol behavior, service, and other statistics about connections with the same destination host as the current connection over the past 2 seconds, and same-service features, which check only connections that have been connected to the same server in the past 2 seconds.

Content feature: For cyberattacks, R2L and U2R attacks do not have frequent sequential intrusion patterns like DoS and probing attacks. Because DoS and probe attacks involve many connections in a short period, R2L and U2R attacks typically involve only one connection. Thus, more in-depth features are needed to detect these kinds of attacks, e.g., the number of unsuccessful login attempts, which is in the payload part of the packet. Such features are referred to as content features.

Biglar Beigi et al. [85] also discussed distinct flow-based features thoroughly:

- **Source and destination IP addresses:** One of the primary advantages of their use is the possibility of quantifying the number of distinct connections. However, it does not provide a definitive conclusion as a feature.
- **Source and destination port:** Botnet traffic was often recognized by source and destination ports, but this is not the case anymore due to the frequent change of source and destination ports.
- **Protocol:** It reduced the volume of flows that needed to be processed by filtering out non-related traffic. Furthermore, a certain protocol might be an indication of malicious traffic, such as Internet Relay Chat (IRC), because it is rarely used for non-malicious purposes.
- **Duration:** This feature could help identify malicious traffic. For example, most botnets initiate a single-way connection and maintain very short communication sessions before attempting a multidirectional connection.
- **First packet length:** In addition to revealing the underlying protocol's characteristics, the first packet in the flow is useful in detecting malicious traffic.
- **Flow size features:** Communication patterns are mainly intended to be represented by such features.
- **Ratio between the number of incoming packets over the number of outgoing packets:** It has been noted in studies examining the breakdown of Internet traffic that there is an even distribution between inbound and outbound traffic for various protocols. This would help identify further traffic characterization [86].
- **Packet exchange:** The number of network packets exchanged between two entities could help to identify malicious traffic. For instance, bots are always trying to keep communication alive by sending many packets.
- **Reconnect:** Some malicious entities might try to disconnect and reconnect again to break the detection process using feature extraction. To overcome this problem, a certain number of reconnections are only allowed within a certain interval of time.
- **Number and percentage of small or null packets exchanged:** Studies have shown that certain malicious traffic might exchange small network packets. For instance, IRC hosts exchange small messages with the C&C server [87][88].
- **Average packet or bits per second, average inter-arrival time of packets:** This feature helps identify similar communication patterns.

Different network features are useful for certain applications. We have seen some useful network features in the context of malicious traffic. In the context of IoT device classification, Sivanathan et al. [89] extracted DNS queries, NTP queries, port numbers, and cipher suits for each device within the testbed they developed and used these features to identify the devices in the network. This shows how different features can be used for different applications. As a final thought, extracting more features does not always guarantee a high accuracy anomaly detection rate, according to

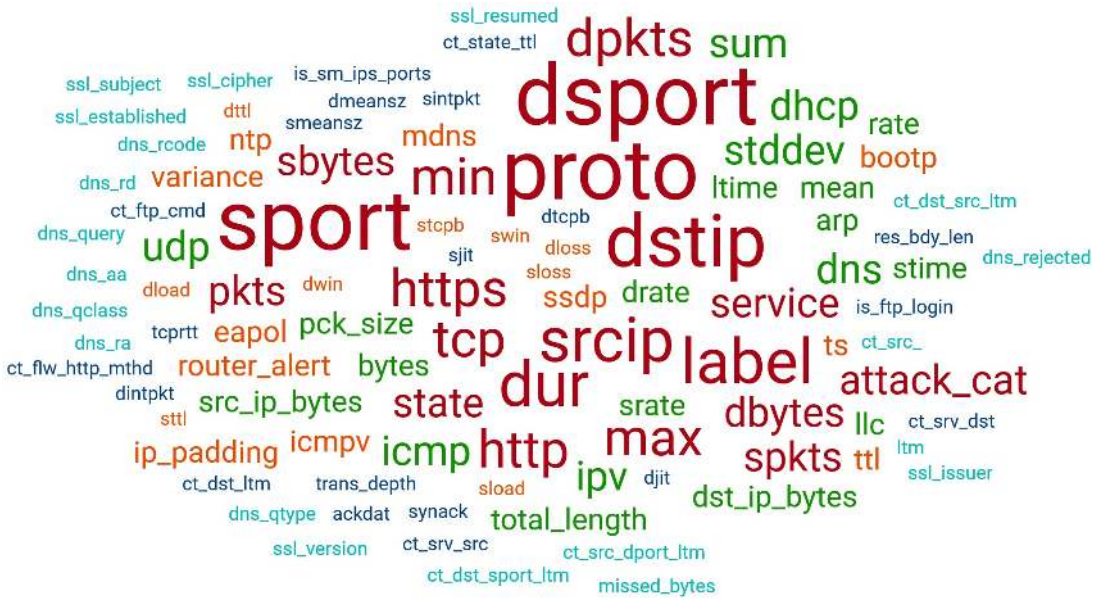

Figure 6: 207 Features extracted from 10 different datasets that show the most common extracted features in big font size, while the least extracted features are in small font size.

Revathi & Malathi [90]. In their paper, they applied RF, naive Bayes, and SVM, among other classifiers, one with 41 features and another with only 13 features, where they used a correlation feature selection (CFS) subset to filter down the features. The result was that some classifiers had a high accuracy detection rate with fewer features. For instance, the RF classifier had an accuracy of 98.9% with only 13 features, whereas it achieved an accuracy of 97.6% with 41 features in detecting probe attacks. Figure 6 shows the most common features extracted from the following datasets: UNSW-NB15 [35], CIC IoT 2022 [53], Bot-IoT [75], MQTT-IoT-IDS2020 [78], DAD [81], TON_IoT [83], CIC IoT 2023 [91], IoT-deNAT [92], IOT Sentinel [93], IoTSense [94].

7.6.2 Statistical Data Analysis

An efficient way to predict and understand IoT device behavior is through mathematical modeling. Using proven approach techniques, it is essential to mathematically model the real environment to generate a dataset as realistic as possible [81]. The benefit of using a statistical-based analysis method is that it analyzes features individually, which eliminates the feature redundancy issue. Li et al. [95] discussed several statistical-based analysis methods that are summarized along with their formulas in Table 8:

- **Low variance:** This method filters out features with low variance.
- **T-score [96]:** It is used for binary classification to determine whether a feature causes a statistically significant difference between two classes.
- **Chi-square score [97]:** This method helps to determine how independent a feature is of the class label.
- **Gini index [98]:** Different classes of samples can be separated using this method.
- **CFS [99]:** It evaluates how a certain feature fits in relation to the data.

8 Commonly Used Tools and Techniques

In this section, we discuss several commonly used tools in the context of developing a testbed and creating a dataset that are summarized in Table 9.

8.1 Simulation

Moustafa & Slay [35] used a hardware network simulation called IXIA PerfectStorm (`keysight.com/gb/en/products/network-test/network-test-hardware/perfectstorm.html`) for the simulation of network traffic and cyberattacks. According to their documentation, combining multiple simultaneous wired and wireless users with

Table 9: Summary of Commonly Used Tools and Techniques

Name	Purpose	References
Argus	Network Feature Extraction	[100][101][82][14][75]
Bro-IDS	Network Analyzer	[14][82]
Tcpdump	Packet Capture & Analyzer	[102][103][75][82][104][105][89][106]
Dpkt	Network Feature Extraction	[53][75][107][82]
IXIA PerfectStorm	Hardware Network Simulator	[14][108][82]
MATLAB	Programming & Numeric Computing	[109][110][111][112][113][114][106] [114][112][115]
Nmap	Network Scanner	[116][43][117][53][118][75][103][119][10]
Hydra	Password Cracker	[75][53][119]
Mirage	Security Analysis	[52]
OpenWRT	OS for Routers	[120][121][105][122][123][89]
CICFlowmeter	Network Feature Extraction	[59][124][125][126][65][127]
LOIC	Network Stress Tool	[53] [103]
TSfresh	Time-Series Feature Extraction	[128][129][130]
Wireshark	Packet Capture & Analyzer	[131][43][107] [10] [117][132]
Tshark	Packet Capture & Analyzer	[4][100] [75][133][133]
PyTorch	Implementing AD Algorithm	[65][134][69][62][30]
PySyft	Implementing AD Algorithm	[65][134][69][62]

both application traffic and current attacks is possible with PerfectStorm Fusion. Furthermore, thousands of real-life end-user environments are simulated and tested using stateful applications and malicious traffic such as DoS and botnets. The workloads include data, video, voice, storage, and network applications.

8.2 Network Feature Extraction

We have previously discussed how important extracting features is for a dataset. Similarly, Frazão et al. [80] extracted network features using MATLAB (mathworks.com), where they extracted 68 network features, including packet timestamps, inter-packet arrival times, binary features defining which protocols were involved, and every field of the Ethernet, ARP, IP, ICMP, UDP, TCP and MODBUS over TCP headers. MATLAB can also be used to simulate an IoT environment using MATLAB Simulink features as presented by Al Farooq et al. [135]. According to MATLAB's website, both the model-based design and multi-domain simulation are supported by Simulink. Argus (`openargus.org`) is another network extraction tool that was used by Koroniotis et al. [75] to extract the record start time, record last time, standard deviation of aggregated records, class label: 0 for normal traffic, 1 for attack traffic, and traffic category among other network features. Likewise, Moustafa & Slay [35] extracted three categories of network features: basic, content, and time using Argus. In addition to auditing IP traffic, Argus provides network activity audit technology for all network traffic, according to their website. Along with Argus, Moustafa & Slay [35] used a network extraction tool called Bro-IDS (`zeek.org`) to extract network features and match the extracted features with the extracted features from Argus. Their website states that Bro-IDS can be configured to act as an IDS and detect well-known attacks. It is also important to know that Bro-IDS has been renamed Zeek. Dpkt (`dpkt.readthedocs.io`) is another feature extraction tool written in Python for creating and parsing network packets, which supports basic TCP/IP protocols. Dadkhah et al. [53] extracted 48 features from each device in their experiment that include packet size, protocol, epoch timestamp, and Ethernet frame size, among other features, using dpkt. According to Teh et al. [68], Tsfresh (`tsfresh.readthedocs.io/en/latest/`) is a widely used ML library for time series feature extraction that is written in Python. Ouyang et al. [129] also used Tsfresh in the context of power consumption to extract features from the power consumption data of 54,174 customers. Furthermore, Liu et al. [128] used Tsfresh along with LSTM to present a sensor fault classification method that is based on feature extraction.

8.3 Network Capture and Analyzing

Wireshark (`wireshark.org`) is a well-known useful tool that is intended to capture and analyze network traffic. It supports hundreds of protocols to be analyzed and provides online and offline analyzing capabilities. Its interface is easy to read, and network capture can be filtered using port numbers, protocols, and IP addresses, among other attributes. Ren et al. [136] used Wireshark to capture network traffic in their project. According to their paper, Wireshark does not support all encrypted protocols. Tshark (`wireshark.org/docs/man-pages/tshark.html`) is the terminal version of Wireshark, which becomes handy in case the use of GUI is not feasible. Similarly, TCPDump (`tcpdump.org`) is another packet analyzer and network capture tool that is widely used nowadays. Hamza et al. [105] captured local and remote network traffic consisting of malicious and benign traffic for 16 days using TCPDump and stored pcap files on a hard disk that is attached to the network gateway. Similarly, CICFlowmeter (`unb.ca/cic`) is a network analyzer

and generator that the Canadian Institute for Cybersecurity developed. It can generate bidirectional traffic and can capture up to 80 different features, such as duration and length of packets. Additionally, a user can select what features to be captured and can add new features. CICFlowmeter has been used by Ullah & Mahmoud [137] to extract network features from pcap files of five different datasets.

8.4 Router Management

To maximize the experience of having full control of routers and access points, Bhatt & Morais [121] configured the router they used in their research using OpenWRT (`openwrt.org`). OpenWRT is an embedded Linux operating system whose file system is fully writable and has a package management feature rather than creating a single, static firmware, as mentioned on their website. Additionally, Pessoa & Duarte-Figueiredo [138] used OpenWRT to change the firmware of the TPLink1043nd v2.1 access point to enable OpenFlow connections.

8.5 Cyber Anomalies

To perform cyber anomalies and test the security of their testbed, Dadkhah et al. [53] performed DoS attack using LOIC, which is a network stress tool that performs cyberattacks. Likewise, Bhuyan et al. [103] used LOIC to generate a distributed DoS attack. Security analysis of wireless communications is made easier with Mirage, a powerful and modular framework that is based on Romain Cayre's research on IoT security. The tool provides hackable wireless protocol stacks such as Zigbee, BLE, and Wi-Fi. Using Mirage, Gimenez et al. [52] performed different types of attacks such as DoS, injection, and probe. Hydra is another cyber anomaly tool that helps crack passwords by performing dictionary attacks. Similarly, Koroniotis et al. [75] performed a dictionary attack on their SSH service. Furthermore, Rose et al. [119] created malicious pcap files performing different attacks, using Hydra to perform brute force attacks to test their framework. Another well-known security tool is Nmap, an open-source security auditing and network discovery tool. It can find hosts within a network, services each host is running, and ports available. Furthermore, Nmap can be used to perform cyberattacks by running malicious scripts through the Nmap scripting engine. As an example of using Nmap, Siboni et al. [43] used Nmap to perform a network mapping attack. Even though Nmap can work on different operating systems, they modified Nmap to work on Android as the experiment was conducted on a Sony smartwatch. For vulnerability scanning, Tekeoglu & Tosun [107] used Nmap to scan for vulnerability on a FireTV stick, which shows severe threats. First, a vulnerability was found in the FireTV stick server running on port 49986 with a 10 (out of 10) severity score against `Format String Attack on URI`. Another vulnerability was also found with a 7.5 (out of 10) score of severity on the same port called `CERN httpd CGI name heap overflow`, which allows an attacker to stop a code on the server or even run a malicious code on the server.

8.6 Anomaly Detection Algorithms Implementation

Applying anomaly detection models is a part of the anomaly detection process; thus, we also list tools that help implement an anomaly detection model. Sater & Hamza [69] used PyTorch (`pytorch.org`) to implement the stacked LSTM as an anomaly detection algorithm. PyTorch is an open-source ML framework that is used for DL utilizing GPUs and CPUs. Similarly, PySyft (`openmined.github.io/PySyft/index.html`) is another open-source tool that is used to provide secure and private DL in Python. Mothukuri et al. [65] used PySyft along with PyTorch to implement their FL approach.

9 Evaluation Metrics

For every anomaly detection technique presented, there must be an evaluation criterion that gives an indication of how efficient an anomaly detection technique is. In this section, we present the most common evaluation metrics that are used to evaluate different anomaly detection techniques as summarized in Table 10. Furthermore, we summarize value interpretations that yield a quality anomaly detection technique in Figure 7 and formulas of discussed evaluation metrics in Table 11

- **TP** is when an anomaly is correctly predicted by a model.
- **TN** is when the absence of an anomaly is correctly predicted by a model.
- **FP** is when an anomaly is incorrectly predicted by a model.
- **FN** is when the absence of an anomaly is incorrectly predicted by a model.
- **Accuracy** measures how accurate a model's prediction is. Accuracy is inefficient if a dataset is imbalanced [139].

Table 10: Anomaly Detection Evaluation Metrics Summary

Paper	TP	TN	FP	FN	Accuracy	Precision	Recall	F1-Score	Threshold	Runtime	ROC	AUC	EER	MCC	PPV	MAE	MSE	RMSE	BA
Sikder et al. [45]	●	●	●	●	●	●	●	●	○	○	○	○	○	○	○	○	○	○	○
Ullah & Mahmoud [59]	●	●	●	●	●	●	●	●	○	○	○	○	○	○	○	○	○	○	○
Kang et al. [54]	○	○	○	○	○	○	○	○	●	○	○	○	○	○	○	○	○	○	○
Hosseini et al. [60]	●	●	●	●	●	●	●	●	○	○	○	○	○	○	○	○	○	○	○
Li et al. [61]	●	●	●	●	○	○	○	○	●	○	●	●	●	○	○	○	○	○	○
Liu et al. [62]	●	●	●	●	●	○	○	○	●	○	○	○	○	○	○	○	○	○	○
Mudgerikar et al. [63]	●	●	●	●	●	●	●	●	○	○	●	○	○	○	○	○	○	○	○
Djenouri et al. [64]	●	●	●	●	●	●	●	●	○	●	○	○	○	○	○	○	○	○	○
Mothukuri et al. [65]	●	●	●	●	●	●	●	●	○	○	○	○	○	○	○	○	○	○	○
Shafi et al. [66]	●	●	●	●	●	●	●	●	○	○	●	○	○	○	○	○	○	○	○
Yahyaoui et al. [67]	●	●	●	●	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Gimenez et al. [52]	●	●	●	●	●	●	●	○	○	○	○	○	○	○	○	○	○	○	○
Heartfield et al. [4]	●	●	●	●	●	●	●	●	○	○	●	●	○	○	●	○	○	○	○
Teh et al. [68]	●	●	●	●	○	○	○	○	●	○	○	○	○	●	○	○	○	○	○
Sater & Hamza [69]	●	●	●	●	○	●	●	●	○	○	●	●	○	●	○	●	●	●	●

○ Has not been used, ● Has been used.

- **Precision** measures the ratio of predicted anomalies to the total data samples [139].
- **Recall** measures how many anomalies (TP) were classified by the model [139].
- **F1-Score** measures the accuracy of a model, using precision and recall, especially when the dataset is imbalanced [60].
- **Threshold** is a value that distinguishes between two binary classes, in our case, normal or abnormal. Less than a threshold is a normal instance, whereas any value over it is an anomaly.
- **Runtime** is the time it takes a model to completely run and finish, which is also known as computational time.
- **ROC** provides a graphical representation of a model's performance using the TPR, represented as a y-axis in a graph, and FP rate (FPR), represented as an x-axis in a graph [139].
- **AUC** measures the area under the ROC curve. Essentially, it measures how much area under the ROC has been covered [139].
- **EER** is an intersection point between the TPR and FPR on the ROC graph where FPR and FN rate (FNR) values are equal [139].
- **Matthews Correlation Coefficient (MCC)** measures how accurate a model is in terms of anomaly prediction taking into consideration all four evaluation metrics: TP, TF, FP, and FN. Furthermore, MCC yields a high score if most of both negative and positive data samples are correctly predicted [140].
- **Predictive positive value (PPV)** measures the proportion of positive anomalies that are actually positive [141]
- **Mean absolute error (MAE)** evaluates how closely the predicted anomalies match the actual anomalies [139].
- **Mean squared error (MSE)** is similar to MAE, but it is more sensitive to anomalies because errors are squared [139].
- **Root mean squared error (RMSE)** is a measure of the average of squared errors squared [139].
- **Balanced accuracy (BA)** measures how accurate the prediction of a model is when the dataset is imbalanced [69].

10 Conclusion

In the end, we reviewed and presented guidelines based on scientific background to design a testbed, choose a testbed layout, select users and devices, and define the experimental condition for the testbed. Additionally, different anomaly-creation methods were presented. We have also reviewed multiple anomaly detection mechanisms in terms of the detection algorithm used, whether it was tested on a testbed or a dataset, and the types of anomalies generated.

Furthermore, we discussed the type of infrastructure to create a dataset and what network configuration to choose because different wireless communication protocols might need different hubs to manage them; hence, the budget of the implementation will go up. In terms of data collection, we provided six different statuses in which data are collected during power, idle, interaction, scenarios, active, and attacks. To generate the data, different traffic scenarios must be

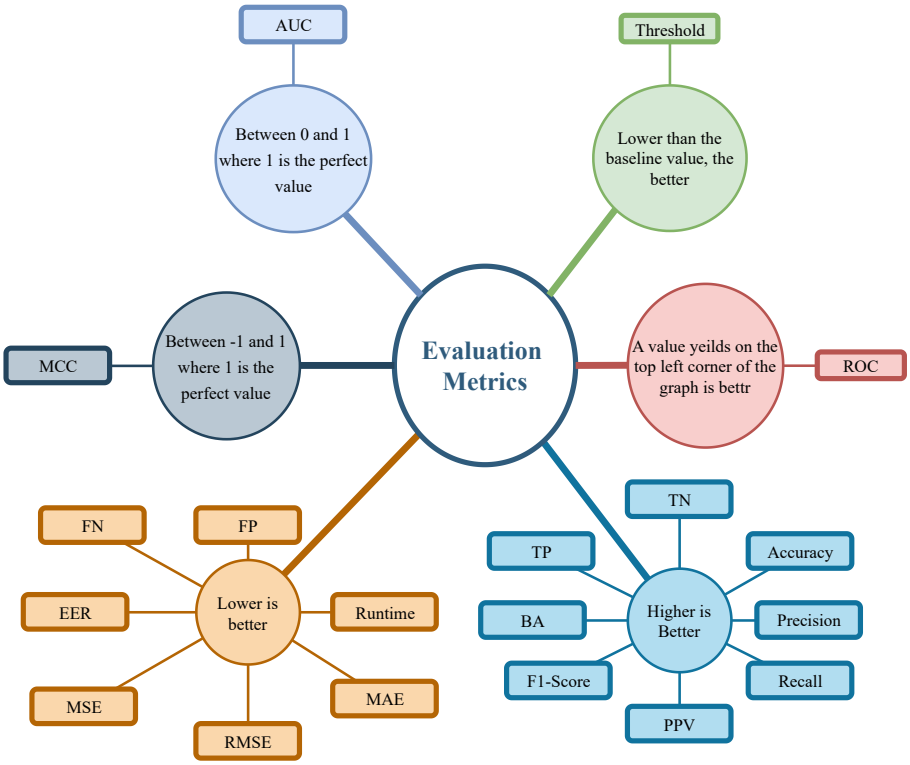

Figure 7: Evaluation metrics and their value interpretations, where it shows whether a lower value or a higher value yields a quality anomaly detection technique.

Table 11: Anomaly Detection Evaluation Metrics Formulas

Evaluation Metric	Formula	No.
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$	(1)
Precision	$\frac{TP}{TP+FP}$	(2)
Recall	$\frac{TP}{TP+FN}$	(3)
F1-Score	$2 \times \frac{Precision \times Recall}{Precision+Recall}$	(4)
ROC	$TPR = \frac{TP}{TP+FN} \quad FPR = \frac{FP}{FP+TN}$	(5)
MCC	$\frac{(TP \times TN)-(FP \times FN)}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$	(6)
PPV	$\frac{TP}{TP+FP}$	(7)
MAE	$\frac{1}{n}\sum_{i=1}^{n}\lvert y_i - \hat{y}_i \rvert$	(8)
MSE	$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$	(9)
RMSE	$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$	(10)
BA	$\frac{TPR+TNR}{2}$	(11)

considered and followed, as we explained with examples of smart home scenarios that might differ depending on the goal of the dataset. Additionally, a different capture duration was presented to assess the minimum requirement for capturing data for a dataset. Dataset analysis is an important aspect of the dataset creation process, which consists of feature extraction and statistical analysis. Several network feature types were presented with examples of distinct network features and their descriptions. Statistical-based analysis methods with low variance, T-score, chi-squared score, Gini index, and CFS were discussed in detail.

A set of useful tools, in terms of capturing and analyzing network traffic such as Wireshark, extracting network features such as CICFlowmeter, and performing cyberattacks such as Mirage, was presented along with their websites and references. We also highlighted the most common anomaly detection evaluation metrics used in the literature.

Acknowledgments

This work has been partially supported by the PETRAS National Centre of Excellence for IoT Systems Cybersecurity, which has been funded by the UK EPSRC under grant number EP/S035362/1.

References

[1] Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, and Moussa Ayyash. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys & Tutorials*, 17(4):2347–2376, 2015.

[2] James Manyika, Michael Chui, Peter Bisson, Jonathan Woetzel, Richard Dobbs, Jacques Bughin, and Dan Aharon. The internet of things mapping the value beyond the hype, Jun 2015.

[3] Mohammad Hasan. State of iot 2022: Number of connected iot devices growing 18

[4] Ryan Heartfield, George Loukas, Anatolij Bezemskij, and Emmanouil Panaousis. Self-Configurable Cyber-Physical Intrusion Detection for Smart Homes Using Reinforcement Learning. *IEEE Transactions on Information Forensics and Security*, 16:1720–1735, 2021. Conference Name: IEEE Transactions on Information Forensics and Security.

[5] Sowmya Ramapatruni, Sandeep Nair Narayanan, Sudip Mittal, Anupam Joshi, and Karuna Joshi. Anomaly detection models for smart home security. In *2019 IEEE 5th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*, pages 19–24, 2019.

[6] Bogdan Copos, Karl Levitt, Matt Bishop, and Jeff Rowe. Is anybody home? inferring activity from smart home network traffic. In *2016 IEEE Security and Privacy Workshops (SPW)*, pages 245–251, 2016.

[7] Vic Barnett and Toby Lewis. *Outliers in statistical data*. Wiley, 1974.

[8] Andrew A. Cook, Göksel Mısırlı, and Zhong Fan. Anomaly detection for iot time-series data: A survey. *IEEE Internet of Things Journal*, 7(7):6481–6494, 2020.

[9] Preetam Jinka and Baron Schwartz. *Anomaly Detection for Monitoring*. O'Reilly Media, Inc., 2015.

[10] Chun Zhu, Weihua Sheng, and Meiqin Liu. Wearable sensor-based behavioral anomaly detection in smart assisted living systems. *IEEE Transactions on Automation Science and Engineering*, 12(4):1225–1234, 2015.

[11] Andrew A. Cook, Göksel Mısırlı, and Zhong Fan. Anomaly Detection for IoT Time-Series Data: A Survey. *IEEE Internet of Things Journal*, 7(7):6481–6494, July 2020. Conference Name: IEEE Internet of Things Journal.

[12] Yassine Himeur, Khalida Ghanem, Abdullah Alsalemi, Faycal Bensaali, and Abbes Amira. Artificial intelligence based anomaly detection of energy consumption in buildings: A review, current trends and new perspectives. *Applied Energy*, 287:116601, April 2021.

[13] L. Erhan, M. Ndubuaku, M. Di Mauro, W. Song, M. Chen, G. Fortino, O. Bagdasar, and A. Liotta. Smart anomaly detection in sensor systems: A multi-perspective review. *Information Fusion*, 67:64–79, March 2021.

[14] Nour Moustafa, Jiankun Hu, and Jill Slay. A holistic review of Network Anomaly Detection Systems: A comprehensive survey. *Journal of Network and Computer Applications*, 128:33–55, February 2019.

[15] Anuroop Gaddam, Tim Wilkin, and Maia Angelova. Anomaly Detection Models for Detecting Sensor Faults and Outliers in the IoT - A Survey. In *2019 13th International Conference on Sensing Technology (ICST)*, pages 1–6, December 2019. ISSN: 2156-8073.

[16] Kelton A. P. da Costa, João P. Papa, Celso O. Lisboa, Roberto Munoz, and Victor Hugo C. de Albuquerque. Internet of Things: A survey on machine learning-based intrusion detection approaches. *Computer Networks*, 151:147–157, March 2019.

[17] Yuan Luo, Ya Xiao, Long Cheng, Guojun Peng, and Danfeng (Daphne) Yao. Deep Learning-based Anomaly Detection in Cyber-physical Systems: Progress and Opportunities. *ACM Computing Surveys*, 54(5):1–36, June 2022.

[18] Elhadj Benkhelifa, Thomas Welsh, and Walaa Hamouda. A Critical Review of Practices and Challenges in Intrusion Detection Systems for IoT: Toward Universal and Resilient Systems. *IEEE Communications Surveys & Tutorials*, 20(4):3496–3509, 2018. Conference Name: IEEE Communications Surveys & Tutorials.

[19] Muhammad Fahim and Alberto Sillitti. Anomaly Detection, Analysis and Prediction Techniques in IoT Environment: A Systematic Literature Review. *IEEE Access*, 7:81664–81681, 2019. Conference Name: IEEE Access.

[20] Ayman Taha and Ali S. Hadi. Anomaly Detection Methods for Categorical Data: A Review. *ACM Computing Surveys*, 52(2):38:1–38:35, May 2019.

[21] Tharindu Fernando, Harshala Gammulle, Simon Denman, Sridha Sridharan, and Clinton Fookes. Deep Learning for Medical Anomaly Detection – A Survey. *ACM Computing Surveys*, 54(7):141:1–141:37, July 2021.

[22] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3), jul 2009.

[23] Kinza Arshad, Rao Faizan Ali, Amgad Muneer, Izzatdin Abdul Aziz, Sheraz Naseer, Nabeel Sabir Khan, and Shakirah Mohd Taib. Deep reinforcement learning for anomaly detection: A systematic review. *IEEE Access*, 10:124017–124035, 2022.

[24] Jordan Frery, Amaury Habrard, Marc Sebban, Olivier Caelen, and Liyun He-Guelton. Efficient top rank optimization with gradient boosting for supervised anomaly detection. In Michelangelo Ceci, Jaakko Hollmén, Ljupčo Todorovski, Celine Vens, and Sašo Džeroski, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 20–35, Cham, 2017. Springer International Publishing.

[25] Yingjie Tian, Mahboubeh Mirzabagheri, Seyed Mojtaba Hosseini Bamakan, Huadong Wang, and Qiang Qu. Ramp loss one-class support vector machine; a robust and effective approach to anomaly detection problems. *Neurocomputing*, 310:223–235, 2018.

[26] Thomas Schlegl, Philipp Seeböck, Sebastian M. Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In Marc Niethammer, Martin Styner, Stephen Aylward, Hongtu Zhu, Ipek Oguz, Pew-Thian Yap, and Dinggang Shen, editors, *Information Processing in Medical Imaging*, pages 146–157, Cham, 2017. Springer International Publishing.

[27] Guansong Pang, Anton Hengel, Chunhua Shen, and Longbing Cao. *Deep Reinforcement Learning for Unknown Anomaly Detection*. arXiv, September 2020.

[28] Xiaojin Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.

[29] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P. Trevino, Jiliang Tang, and Huan Liu. Feature selection: A data perspective. *ACM Comput. Surv.*, 50(6), dec 2017.

[30] Jiuqi Elise Zhang, Di Wu, and Benoit Boulet. Time Series Anomaly Detection via Reinforcement Learning-Based Model Selection, July 2022. arXiv:2205.09884 [cs].

[31] Lukas Ruff, Robert A. Vandermeulen, Nico Görnitz, Alexander Binder, Emmanuel Müller, Klaus-Robert Müller, and Marius Kloft. Deep Semi-Supervised Anomaly Detection, February 2020. arXiv:1906.02694 [cs, stat].

[32] Guansong Pang, Chunhua Shen, and Anton van den Hengel. Deep anomaly detection with deviation networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 353–362, New York, NY, USA, 2019. Association for Computing Machinery.

[33] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation-based anomaly detection. *ACM Trans. Knowl. Discov. Data*, 6(1), mar 2012.

[34] Guansong Pang, Longbing Cao, Ling Chen, and Huan Liu. Learning Representations of Ultrahigh-dimensional Data for Random Distance-based Outlier Detection. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2041–2050, July 2018. arXiv:1806.04808 [cs, stat].

[35] Nour Moustafa and Jill Slay. Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In *2015 Military Communications and Information Systems Conference (MilCIS)*, pages 1–6, 2015.

[36] W. Schiffmann, M. Joost, and R. Werner. Synthesis and Performance Analysis of Multilayer Neural Network Architectures, 1992.

[37] W Schiffmann, M Joost, and R Werner. Optimization of the Backpropagation Algorithm for Training Multilayer Perceptrons. page 36, 1994.

[38] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge L Reyes-Ortiz. A Public Domain Dataset for Human Activity Recognition Using Smartphones. *Computational Intelligence*, page 6, 2013.

[39] Jock Blackard and Denis Dean. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and Electronics in Agriculture*, 24:131–151, 12 1999.

[40] Robert Müller. Towards Anomaly Detection in Reinforcement Learning. *AAMAS '22: Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, page 5, 2022.

[41] Barbara Kitchenham. Procedures for performing systematic reviews. *Keele, UK, Keele University*, 33(2004):1–26, 2004.

[42] Jie Jiang, Riccardo Pozza, Nigel Gilbert, and Klaus Moessner. Makesense: an iot testbed for social research of indoor activities. *ACM Transactions on Internet of Things*, 1(3):1–25, 2020.

[43] Shachar Siboni, Asaf Shabtai, Nils O. Tippenhauer, Jemin Lee, and Yuval Elovici. Advanced Security Testbed Framework for Wearable IoT Devices. *ACM Transactions on Internet Technology*, 16(4):1–25, December 2016.

[44] Sajal Bhatia, Desmond Schmidt, George Mohay, and Alan Tickle. A framework for generating realistic traffic for Distributed Denial-of-Service attacks and Flash Events. *Computers & Security*, 40:95–107, February 2014.

[45] Amit Kumar Sikder, Leonardo Babun, and A Selcuk Uluagac. Aegis+ a context-aware platform-independent security framework for smart home systems. *Digital Threats: Research and Practice*, 2(1):1–33, 2021.

[46] Xinyu Lei, Guan-Hua Tu, Alex X. Liu, Chi-Yu Li, and Tian Xie. The Insecurity of Home Digital Voice Assistants - Vulnerabilities, Attacks and Countermeasures. In *2018 IEEE Conference on Communications and Network Security (CNS)*, pages 1–9, May 2018.

[47] Diane J. Cook, Aaron S. Crandall, Brian L. Thomas, and Narayanan C. Krishnan. CASAS: A Smart Home in a Box. *Computer*, 46(7):62–69, July 2013. Conference Name: Computer.

[48] Waheb A. Jabbar, Tee Kok Kian, Roshahliza M. Ramli, Siti Nabila Zubir, Nurthaqifah S. M. Zamrizaman, Mohammed Balfaqih, Vladimir Shepelev, and Soltan Alharbi. Design and Fabrication of Smart Home With Internet of Things Enabled Automation System. *IEEE Access*, 7:144059–144074, 2019. Conference Name: IEEE Access.

[49] Mauro Piva, Andrea Coletta, Gaia Maselli, and John A. Stankovic. Environment-driven Communication in Battery-free Smart Buildings. *ACM Transactions on Internet of Things*, 2(2):1–30, May 2021.

[50] Sunny Behal and Krishan Kumar. Detection of DDoS attacks and flash events using information theory metrics–An empirical investigation. *Computer Communications*, 103:18–28, May 2017.

[51] Jingjing Ren, Daniel J Dubois, David Choffnes, Anna Maria Mandalari, Roman Kolcun, and Hamed Haddadi. Information exposure from consumer iot devices: A multidimensional, network-informed measurement approach. In *Proceedings of the Internet Measurement Conference*, pages 267–279, 2019.

[52] Pierre-Francois Gimenez, Jonathan Roux, Eric Alata, Guillaume Auriol, Mohamed Kaaniche, and Vincent Nicomette. RIDS: Radio Intrusion Detection and Diagnosis System for Wireless Communications in Smart Environment. *ACM Transactions on Cyber-Physical Systems*, 5(3):1–1, July 2021.

[53] Sajjad Dadkhah, Hassan Mahdikhani, Priscilla Kyei Danso, Alireza Zohourian, Kevin Anh Truong, and Ali A. Ghorbani. Towards the Development of a Realistic Multidimensional IoT Profiling Dataset. In *2022 19th Annual International Conference on Privacy, Security & Trust (PST)*, pages 1–11, August 2022.

[54] Kai Kang, Lijie Xu, Wei Wang, Guoquan Wu, Jun Wei, Wei Shi, and Jizhong Li. A Hierarchical Automata Based Approach for Anomaly Detection in Smart Home Devices. In *2020 International Conferences on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics)*, pages 1–8, November 2020.

[55] Abdel Mlak Said, Aymen Yahyaoui, and Takoua Abdellatif. Efficient Anomaly Detection for Smart Hospital IoT Systems. *Sensors*, 21(4):1026, January 2021. Number: 4 Publisher: Multidisciplinary Digital Publishing Institute.

[56] Jelena Mirkovic, Alefiya Hussain, Sonia Fahmy, Peter Reiher, and Roshan K. Thomas. Accurately Measuring Denial of Service in Simulation and Testbed Experiments. *IEEE Transactions on Dependable and Secure Computing*, 6(2):81–95, April 2009. Conference Name: IEEE Transactions on Dependable and Secure Computing.

[57] T. Benzel, R. Braden, D. Kim, C. Neuman, A. Joseph, K. Sklower, R. Ostrenga, and S. Schwab. Experience with deter: a testbed for security research. In *2nd International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities, 2006. TRIDENTCOM 2006.*, pages 10 pp.–388, 2006.

[58] J. Lundström, W. Ourique De Morais, and M. Cooney. A holistic smart home demonstrator for anomaly detection and response. In *2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, pages 330–335, March 2015.

[59] Imtiaz Ullah and Qusay H. Mahmoud. A Framework for Anomaly Detection in IoT Networks Using Conditional Generative Adversarial Networks. *IEEE Access*, 9:165907–165931, 2021. Conference Name: IEEE Access.

[60] Sayed Saeed Hosseini, Kodjo Agbossou, Sousso Kelouwani, Alben Cardenas, and Nilson Henao. A practical approach to residential appliances on-line anomaly detection: A case study of standard and smart refrigerators. *IEEE Access*, 8:57905–57922, 2020.

[61] Ruoyu Li, Qing Li, Jianer Zhou, and Yong Jiang. Adriot: An edge-assisted anomaly detection framework against iot-based network attacks. *IEEE Internet of Things Journal*, 9(13):10576–10587, 2022.

[62] Yi Liu, Sahil Garg, Jiangtian Nie, Yang Zhang, Zehui Xiong, Jiawen Kang, and M. Shamim Hossain. Deep Anomaly Detection for Time-Series Data in Industrial IoT: A Communication-Efficient On-Device Federated Learning Approach. *IEEE Internet of Things Journal*, 8(8):6348–6358, April 2021. Conference Name: IEEE Internet of Things Journal.

[63] Anand Mudgerikar, Puneet Sharma, and Elisa Bertino. Edge-Based Intrusion Detection for IoT devices. *ACM Transactions on Management Information Systems*, 11(4):1–21, December 2020.

[64] Youcef Djenouri, Djamel Djenouri, Asma Belhadi, Gautam Srivastava, and Jerry Chun-Wei Lin. Emergent Deep Learning for Anomaly Detection in Internet of Everything. *IEEE Internet of Things Journal*, pages 1–1, 2021. Conference Name: IEEE Internet of Things Journal.

[65] Viraaji Mothukuri, Prachi Khare, Reza M. Parizi, Seyedamin Pouriyeh, Ali Dehghantanha, and Gautam Srivastava. Federated-Learning-Based Anomaly Detection for IoT Security Attacks. *IEEE Internet of Things Journal*, 9(4):2545–2554, February 2022. Conference Name: IEEE Internet of Things Journal.

[66] Qaisar Shafi, Abdul Basit, Saad Qaisar, Abigail Koay, and Ian Welch. Fog-Assisted SDN Controlled Framework for Enduring Anomaly Detection in an IoT Network. *IEEE Access*, 6:73713–73723, 2018. Conference Name: IEEE Access.

[67] Aymen Yahyaoui, Takoua Abdellatif, Sami Yangui, and Rabah Attia. READ-IoT: Reliable Event and Anomaly Detection Framework for the Internet of Things. *IEEE Access*, 9:24168–24186, 2021. Conference Name: IEEE Access.

[68] Hui Yie Teh, Kevin I-Kai Wang, and Andreas W. Kempa-Liehr. Expect the Unexpected: Unsupervised Feature Selection for Automated Sensor Anomaly Detection. *IEEE Sensors Journal*, 21(16):18033–18046, August 2021. Conference Name: IEEE Sensors Journal.

[69] Raed Abdel Sater and A. Ben Hamza. A Federated Learning Approach to Anomaly Detection in Smart Buildings. *ACM Transactions on Internet of Things*, 2(4):1–23, November 2021.

[70] Kai Kang, Lijie Xu, Wei Wang, Guoquan Wu, Jun Wei, Wei Shi, and Jizhong Li. A hierarchical automata based approach for anomaly detection in smart home devices. In *2020 International Conferences on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics)*, pages 1–8. IEEE, 2020.

[71] Anand Mudgerikar, Puneet Sharma, and Elisa Bertino. Edge-based intrusion detection for iot devices. *ACM Trans. Manage. Inf. Syst.*, 11(4), oct 2020.

[72] Yin Minn Pa Pa, Shogo Suzuki, Katsunari Yoshioka, Tsutomu Matsumoto, Takahiro Kasama, and Christian Rossow. Iotpot: Analysing the rise of iot compromises. In *Proceedings of the 9th USENIX Conference on Offensive Technologies*, WOOT'15, page 9, USA, 2015. USENIX Association.

[73] Wenke Lee, Salvatore J. Stolfo, and Kui W. Mok. Mining in a data-flow environment: Experience in network intrusion detection. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '99, page 114–124, New York, NY, USA, 1999. Association for Computing Machinery.

[74] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani. A detailed analysis of the kdd cup 99 data set. In *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pages 1–6, 2009.

[75] Nickolaos Koroniotis, Nour Moustafa, Elena Sitnikova, and Benjamin Turnbull. Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset. *Future Generation Computer Systems*, 100:779–796, November 2019.

[76] Sebastian Garcia, Agustin Parmisano, and Maria Jose Erquiaga. IoT-23: A labeled dataset with malicious and benign IoT network traffic, jan 2020. More details here https://www.stratosphereips.org/datasets-iot23.

[77] Hyunjae Kang, Dong Hyun Ahn, Gyung Min Lee, Jeong Do Yoo, Kyung Ho Park, and Huy Kang Kim. Iot network intrusion dataset, 2019.

[78] Hanan Hindy, Ethan Bayne, Miroslav Bures, Robert Atkinson, Christos Tachtatzis, and Xavier Bellekens. Machine Learning Based IoT Intrusion Detection System: An MQTT Case Study (MQTT-IoT-IDS2020 Dataset), November 2020. arXiv:2006.15340 [cs].

[79] Ivan Vaccari, Giovanni Chiola, Maurizio Aiello, Maurizio Mongelli, and Enrico Cambiaso. Mqttset, a new dataset for machine learning techniques on mqtt. *Sensors*, 20(22), 2020.

[80] Ivo Frazão, Pedro Henriques Abreu, Tiago Cruz, Hélder Araújo, and Paulo Simões. Denial of Service Attacks: Detecting the Frailties of Machine Learning Algorithms in the Classification Process. In Eric Luiijf, Inga Žutautaitė, and Bernhard M. Hämmerli, editors, *Critical Information Infrastructures Security*, Lecture Notes in Computer Science, pages 230–235, Cham, 2019. Springer International Publishing.

[81] Laura Vigoya, Diego Fernandez, Victor Carneiro, and Fidel Cacheda. Annotated Dataset for Anomaly Detection in a Data Center with IoT Sensors. *Sensors*, 20(13):3745, January 2020. Number: 13 Publisher: Multidisciplinary Digital Publishing Institute.

[82] Nour Moustafa and Jill Slay. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In *2015 Military Communications and Information Systems Conference (MilCIS)*, pages 1–6, November 2015.

[83] Nour Moustafa. A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_iot datasets. *Sustainable Cities and Society*, 72:102994, 2021.

[84] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani. A detailed analysis of the KDD CUP 99 data set. In *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pages 1–6, July 2009. ISSN: 2329-6275.

[85] Elaheh Biglar Beigi, Hossein Hadian Jazi, Natalia Stakhanova, and Ali A. Ghorbani. Towards effective feature selection in machine learning-based botnet detection approaches. In *2014 IEEE Conference on Communications and Network Security*, pages 247–255, October 2014.

[86] Wolfgang John and S. Tafvelin. Differences between In- and Outbound Internet Backbone Traffic. 2007.

[87] David Zhao, Issa Traore, Ali Ghorbani, Bassam Sayed, Sherif Saad, and Wei Lu. Peer to Peer Botnet Detection Based on Flow Intervals. volume 376, June 2012.

[88] Wen-Hwa Liao and Chia-Ching Chang. Peer to Peer Botnet Detection Using Data Mining Scheme. In *2010 International Conference on Internet Technology and Applications*, pages 1–4, August 2010.

[89] Arunan Sivanathan, Hassan Habibi Gharakheili, Franco Loi, Adam Radford, Chamith Wijenayake, Arun Vishwanath, and Vijay Sivaraman. Classifying IoT Devices in Smart Environments Using Network Traffic Characteristics. *IEEE Transactions on Mobile Computing*, 18(8):1745–1759, August 2019. Conference Name: IEEE Transactions on Mobile Computing.

[90] S Revathi and Dr A Malathi. A Detailed Analysis on NSL-KDD Dataset Using Various Machine Learning Techniques for Intrusion Detection. *International Journal of Engineering Research*, 2(12), 2013.

[91] Euclides Carlos Pinto Neto, Sajjad Dadkhah, Raphael Ferreira, Alireza Zohourian, Rongxing Lu, and Ali A. Ghorbani. Ciciot2023: A real-time dataset and benchmark for large-scale attacks in iot environment. *Sensors*, 23(13), 2023.

[92] Yair Meidan, Vinay Sachidananda, Hongyi Peng, Racheli Sagron, Yuval Elovici, and Asaf Shabtai. IoT-deNAT: Outbound flow-based network traffic data of IoT and non-IoT devices behind a home NAT, jul 2020.

[93] Markus Miettinen, Samuel Marchal, Ibbad Hafeez, N. Asokan, Ahmad-Reza Sadeghi, and Sasu Tarkoma. IoT Sentinel: Automated Device-Type Identification for Security Enforcement in IoT. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 2177–2184, June 2017. arXiv:1611.04880 [cs].

[94] Bruhadeshwar Bezawada, Maalvika Bachani, Jordan Peterson, Hossein Shirazi, Indrakshi Ray, and Indrajit Ray. IoTSense: Behavioral Fingerprinting of IoT Devices, April 2018. arXiv:1804.03852 [cs].

[95] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P. Trevino, Jiliang Tang, and Huan Liu. Feature Selection: A Data Perspective. *ACM Computing Surveys*, 50(6):1–45, November 2018.

[96] *Statistics and data analysis in geology, by John C. Davis. With FORTRAN programs, by R.J. Sampson*. New York Wiley & Sons, 1973.

[97] Huan Liu and R. Setiono. Chi2: feature selection and discretization of numeric attributes. pages 388–391, November 1995. ISSN: 1082-3409.

[98] Gini C. W. Variability and mutability, contribution to the study of statistical distributions and relations. studi cconomico-giuridici della r. universita de cagliari (1912). reviewed in : Light, r. j., margolin, b. h. : An analysis of variance for categorical data. *J. American Statistical Association*, 66:534–544, 1971.

[99] Mark A. Hall and Lloyd A. Smith. Feature selection for machine learning: Comparing a correlation-based filter approach to the wrapper. In *Proceedings of the Twelfth International Florida Artificial Intelligence Research Society Conference*, page 235–239. AAAI Press, 1999.

[100] Nikolaos Vakakis, Odysseas Nikolis, Dimosthenis Ioannidis, Konstantinos Votis, and Dimitrios Tzovaras. Cybersecurity in SMEs: The Smart-Home/Office Use Case. In *2019 IEEE 24th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, pages 1–7, September 2019. ISSN: 2378-4873.

[101] Khalid Albulayhi and Frederick T. Sheldon. An Adaptive Deep-Ensemble Anomaly-Based Intrusion Detection System for the Internet of Things. In *2021 IEEE World AI IoT Congress (AIIoT)*, pages 0187–0196, May 2021.

[102] Keyang Yu and Dong Chen. SmartAttack: Open-source Attack Models for Enabling Security Research in Smart Homes. In *2020 11th International Green and Sustainable Computing Workshops (IGSC)*, pages 1–8, October 2020.

[103] Monowar H Bhuyan, Dhruba K Bhattacharyya, and Jugal K Kalita. Towards Generating Real-life Datasets for Network Intrusion Detection. *International Journal of Network Security*, 2015.

[104] Jonathan White and Phil Legg. Unsupervised One-Class Learning for Anomaly Detection on Home IoT Network Devices. In *2021 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)*, pages 1–8, June 2021.

[105] Ayyoob Hamza, Hassan Habibi Gharakheili, Theophilus A. Benson, and Vijay Sivaraman. Detecting Volumetric Attacks on loT Devices via SDN-Based Monitoring of MUD Activity. In *Proceedings of the 2019 ACM Symposium on SDN Research*, pages 36–48, San Jose CA USA, April 2019. ACM.

[106] Mauro Conti, Denis Donadel, and Federico Turrin. A Survey on Industrial Control System Testbeds and Datasets for Security Research. *IEEE Communications Surveys Tutorials*, 23(4):2248–2294, 2021. Conference Name: IEEE Communications Surveys Tutorials.

[107] Ali Tekeoglu and Ali Şaman Tosun. A Testbed for Security and Privacy Analysis of IoT Devices. In *2016 IEEE 13th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pages 343–348, October 2016. ISSN: 2155-6814.

[108] W. Haider, J. Hu, J. Slay, B. P. Turnbull, and Y. Xie. Generating realistic intrusion detection system dataset based on fuzzy qualitative modeling. *Journal of Network and Computer Applications*, 87:185–192, 2017.

[109] Yi Wang, Brandon Saez, Joseph Szczechowicz, Jonathan Ruisi, Tyler Kraft, Stephen Toscano, Zachary Vacco, and Kervins Nicolas. A smart campus internet of things framework. In *2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*, pages 498–503, October 2017.

[110] Minh Pham, Dan Yang, and Weihua Sheng. A Sensor Fusion Approach to Indoor Human Localization Based on Environmental and Wearable Sensors. *IEEE Transactions on Automation Science and Engineering*, 16(1):339–350, January 2019. Conference Name: IEEE Transactions on Automation Science and Engineering.

[111] Christopher Osiegbu, Seifemichael B. Amsalu, Fatemeh Afghah, Daniel Limbrick, and Abdollah Homaifar. Design and Implementation of an Autonomous Wireless Sensor-Based Smart Home. In *2015 24th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–7, August 2015. ISSN: 1095-2055.

[112] Rijad Alisic, Marco Molinari, Philip E. Paré, and Henrik Sandberg. Ensuring Privacy of Occupancy Changes in Smart Buildings. In *2020 IEEE Conference on Control Technology and Applications (CCTA)*, pages 871–876, August 2020.

[113] Jiwon Choi, Hayoung Jeoung, Jihun Kim, Youngjoo Ko, Wonup Jung, Hanjun Kim, and Jong Kim. Detecting and Identifying Faulty IoT Devices in Smart Home with Context Extraction. In *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 610–621, June 2018. ISSN: 2158-3927.

[114] Quang Phuc Ha, Santanu Metia, and Manh Duong Phung. Sensing Data Fusion for Enhanced Indoor Air Quality Monitoring. *IEEE Sensors Journal*, 20(8):4430–4441, April 2020. Conference Name: IEEE Sensors Journal.

[115] Dali Zhu, Na Pang, Gang Li, Wenjing Rong, and Zheming Fan. WiN: Non-invasive Abnormal Activity Detection Leveraging Fine-Grained WiFi Signals. In *2016 IEEE Trustcom/BigDataSE/ISPA*, pages 744–751, August 2016. ISSN: 2324-9013.

[116] Ragav Sridharan, Rajib Ranjan Maiti, and Nils Ole Tippenhauer. WADAC: Privacy-Preserving Anomaly Detection and Attack Classification on Wireless Traffic. In *Proceedings of the 11th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, pages 51–62, Stockholm Sweden, June 2018. ACM.

[117] Vinay Sachidananda, Shachar Siboni, Asaf Shabtai, Jinghui Toh, Suhas Bhairav, and Yuval Elovici. Let the Cat Out of the Bag: A Holistic Approach Towards Security Analysis of the Internet of Things. In *Proceedings of the 3rd ACM International Workshop on IoT Privacy, Trust, and Security*, pages 3–10, Abu Dhabi United Arab Emirates, April 2017. ACM.

[118] Taotao Li, Zhen Hong, and Li Yu. Machine Learning-based Intrusion Detection for IoT Devices in Smart Home. In *2020 IEEE 16th International Conference on Control Automation (ICCA)*, pages 277–282, October 2020. ISSN: 1948-3457.

[119] Joseph R Rose, Matthew Swann, Gueltoum Bendiab, Stavros Shiaeles, and Nicholas Kolokotronis. Intrusion Detection using Network Traffic Profiling and Machine Learning for IoT. In *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*, pages 409–415, June 2021. ISSN: 2693-9789.

[120] Marcin Szczodrak, Yong Yang, Dave Cavalcanti, and Luca P. Carloni. An open framework to deploy heterogeneous wireless testbeds for Cyber-Physical Systems. In *2013 8th IEEE International Symposium on Industrial Embedded Systems (SIES)*, pages 215–224, June 2013. ISSN: 2150-3117.

[121] Parth Bhatt and Anderson Morais. HADS: Hybrid Anomaly Detection System for IoT Environments. In *2018 International Conference on Internet of Things, Embedded Systems and Communications (IINTEC)*, pages 191–196, December 2018.

[122] Pengfei Peng and An Wang. SmartMon: Misbehavior Detection via Monitoring Smart Home Automations. In *2020 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 327–333, November 2020.

[123] Rahmadi Trimananda, Ali Younis, Bojun Wang, Bin Xu, Brian Demsky, and Guoqing Xu. Vigilia: Securing Smart Home Edge Computing. In *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 74–89, October 2018.

[124] Pedro H. A. D. de Melo, Adriano Araújo Martins de Resende, Rodrigo Sanches Miani, and Pedro Frosi Rosa. Evaluation of one-class algorithms for anomaly detection in home networks. In *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 682–689, November 2021. ISSN: 2375-0197.

[125] Nilesh Vishwasrao Patil, C. Rama Krishna, and Krishan Kumar. SSK-DDoS: distributed stream processing framework based classification system for DDoS attacks. *Cluster Computing*, 25(2):1355–1372, April 2022.

[126] Riccardo Pecori, Amin Tayebi, Armando Vannucci, and Luca Veltri. IoT Attack Detection with Deep Learning Analysis. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, July 2020. ISSN: 2161-4407.

[127] Panagiotis Radoglou-Grammatikis, Ilias Siniosoglou, Thanasis Liatifis, Anastasios Kourouniadis, Konstantinos Rompolos, and Panagiotis Sarigiannidis. Implementation and detection of modbus cyberattacks. In *2020 9th International Conference on Modern Circuits and Systems Technologies (MOCAST)*, pages 1–4, 2020.

[128] Gang Liu, Lili Li, Liangliang Zhang, Qing Li, and S. S. Law. Sensor faults classification for SHM systems using deep learning-based method with Tsfresh features. *Smart Materials and Structures*, 29(7):075005, May 2020. Publisher: IOP Publishing.

[129] Zhiyou Ouyang, Xiaokui Sun, and Dong Yue. Hierarchical Time Series Feature Extraction for Power Consumption Anomaly Detection. In Kang Li, Yusheng Xue, Shumei Cui, Qun Niu, Zhile Yang, and Patrick Luk, editors, *Advanced Computational Methods in Energy, Power, Electric Vehicles, and Their Integration*, Communications in Computer and Information Science, pages 267–275, Singapore, 2017. Springer.

[130] Wei Zhang, Xiaowei Dong, Huaibao Li, Jin Xu, and Dan Wang. Unsupervised detection of abnormal electricity consumption behavior based on feature engineering. *IEEE Access*, 8:55483–55500, 2020.

[131] Bilal Al Baalbaki, Jesus Pacheco, Cihan Tunc, Salim Hariri, and Youssif Al-Nashif. Anomaly Behavior Analysis System for ZigBee in smart buildings. In *2015 IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA)*, pages 1–4, November 2015. ISSN: 2161-5330.

[132] Xiaoyang Zhu, Youakim Badr, Jesus Pacheco, and Salim Hariri. Autonomic Identity Framework for the Internet of Things. In *2017 International Conference on Cloud and Autonomic Computing (ICCAC)*, pages 69–79, September 2017.

[133] Holden Gordon, Christopher Batula, Bhagyashri Tushir, Behnam Dezfouli, and Yuhong Liu. Securing Smart Homes via Software-Defined Networking and Low-Cost Traffic Classification. In *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 1049–1057, July 2021. ISSN: 0730-3157.

[134] Theo Ryffel, Andrew Trask, Morten Dahl, Bobby Wagner, Jason Mancuso, Daniel Rueckert, and Jonathan Passerat-Palmbach. A generic framework for privacy preserving deep learning, November 2018.

[135] Abdullah Al Farooq, Ehab Al-Shaer, Thomas Moyer, and Krishna Kant. IoTC2: A Formal Method Approach for Detecting Conflicts in Large Scale IoT Systems. In *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 442–447, April 2019. ISSN: 1573-0077.

[136] Jingjing Ren, Daniel J. Dubois, David Choffnes, Anna Maria Mandalari, Roman Kolcun, and Hamed Haddadi. Information Exposure From Consumer IoT Devices: A Multidimensional, Network-Informed Measurement Approach. In *Proceedings of the Internet Measurement Conference*, pages 267–279, Amsterdam Netherlands, October 2019. ACM.

[137] Imtiaz Ullah and Qusay H. Mahmoud. An Anomaly Detection Model for IoT Networks based on Flow and Flag Features using a Feed-Forward Neural Network. In *2022 IEEE 19th Annual Consumer Communications Networking Conference (CCNC)*, pages 363–368, January 2022. ISSN: 2331-9860.

[138] Talles Quintão Pessoa and Fátima Duarte-Figueiredo. NodePI : An integrated platform for smart homes. In *2017 IEEE 9th Latin-American Conference on Communications (LATINCOM)*, pages 1–6, November 2017.

[139] Samaneh Aminikhanghahi and Diane J. Cook. A Survey of Methods for Time Series Change Point Detection. *Knowledge and Information Systems*, 51(2):339–367, May 2017.

[140] Davide Chicco, Matthijs J. Warrens, and Giuseppe Jurman. The matthews correlation coefficient (mcc) is more informative than cohen's kappa and brier score in binary classification assessment. *IEEE Access*, 9:78368–78381, 2021.

[141] Thomas F. Monaghan, Syed N. Rahman, Christina W. Agudelo, Alan J. Wein, Jason M. Lazar, Karel Everaert, and Roger R. Dmochowski. Foundational Statistical Principles in Medical Research: Sensitivity, Specificity, Positive Predictive Value, and Negative Predictive Value. *Medicina*, 57(5):503, May 2021.

[142] Hossein Jafari, Xiangfang Li, Lijun Qian, and Yuanzhu Chen. Community based sensing: A test bed for environment air quality monitoring using smartphone paired sensors. In *2015 36th IEEE Sarnoff Symposium*, pages 12–17, September 2015.

[143] Hongfei Xue, Wenjun Jiang, Chenglin Miao, Fenglong Ma, Shiyang Wang, Ye Yuan, Shuochao Yao, Aidong Zhang, and Lu Su. DeepMV: Multi-View Deep Learning for Device-Free Human Activity Recognition. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4(1):1–26, March 2020.

[144] J. Lundström, J. Synnott, E. Järpe, and C. D. Nugent. Smart home simulation using avatar control and probabilistic sampling. In *2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, pages 336–341, March 2015.

[145] Tam Van Nguyen, Jin Gook Kim, and Deokjai Choi. ISS: The Interactive Smart home Simulator. In *2009 11th International Conference on Advanced Communication Technology*, volume 03, pages 1828–1833, February 2009. ISSN: 1738-9445.

[146] Nasser Alshammari, Talal Alshammari, Mohamed Sedky, Justin Champion, and Carolin Bauer. OpenSHS: Open Smart Home Simulator. *Sensors*, 17(5):1003, May 2017. Number: 5 Publisher: Multidisciplinary Digital Publishing Institute.

[147] Julien Bruneau, Wilfried Jouve, and Charles Consel. DiaSim: A parameterized simulator for pervasive computing applications. In *2009 6th Annual International Mobile and Ubiquitous Systems: Networking & Services, MobiQuitous*, pages 1–10, July 2009.

[148] J. Synnott, L. Chen, C.D. Nugent, and G. Moore. The creation of simulated activity datasets using a graphical intelligent environment simulation tool. In *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 4143–4146, August 2014. ISSN: 1558-4615.

[149] Jörg Cassens, Felix Schmitt, Tobias Mende, and Michael Herczeg. CASi – A Generic Context Awareness Simulator for Ambient Systems. In Fabio Paternò, Boris de Ruyter, Panos Markopoulos, Carmen Santoro, Evert van Loenen, and Kris Luyten, editors, *Ambient Intelligence*, Lecture Notes in Computer Science, pages 421–426, Berlin, Heidelberg, 2012. Springer.

[150] Shadan Golestan, Alexandr Petcovici, Ioanis Nikolaidis, and Eleni Stroulia. Simulation-Based Deployment Configuration of Smart Indoor Spaces. In *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, pages 358–363, 2019.

[151] Xianzhong Ding, Wan Du, and Alberto Cerpa. OCTOPUS: Deep Reinforcement Learning for Holistic Smart Building Control. In *Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, pages 326–335, New York NY USA, November 2019. ACM.

[152] Mahmud Hossain, Shahid Noor, Yasser Karim, and Ragib Hasan. IoTbed: A Generic Architecture for Testbed as a Service for Internet of Things-Based Systems. In *2017 IEEE International Congress on Internet of Things (ICIOT)*, pages 42–49, Honolulu, HI, USA, June 2017. IEEE.

[153] Ghada Alsuhli and Ahmed Khattab. A Fog-based IoT Platform for Smart Buildings. In *2019 International Conference on Innovative Trends in Computer Engineering (ITCE)*, pages 174–179, February 2019.

[154] Pradnya Gaonkar, Advait Lonkar, GVK Sasirekha, Jyotsna Bapat, and Debabrata Das. Comfort Management System for Smart Buildings: An Open-Source Scalable Prototype. In *2020 IEEE-HYDCON*, pages 1–5, September 2020.

[155] Driss Benhaddou, Lotanna Afogbuom, Farouk Attia, and Muhammad Anan. Autonomous Living Building: Adapting to Occupant's Behavior. In *2019 15th International Wireless Communications Mobile Computing Conference (IWCMC)*, pages 1803–1808, June 2019. ISSN: 2376-6506.

[156] Quanqing Xu, Zhaozheng He, Zengxiang Li, and Mingzhong Xiao. Building an Ethereum-Based Decentralized Smart Home System. In *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*, pages 1004–1009, December 2018. ISSN: 1521-9097.

[157] Wei Zhang, Yan Meng, Yugeng Liu, Xiaokuan Zhang, Yinqian Zhang, and Haojin Zhu. HoMonit: Monitoring Smart Home Apps from Encrypted Traffic. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1074–1088, Toronto Canada, October 2018. ACM.

[158] Miroslav Bures, Bestoun S. Ahmed, Vaclav Rechtberger, Matej Klima, Michal Trnka, Miroslav Jaros, Xavier Bellekens, Dani Almog, and Pavel Herout. PatrIoT: IoT Automated Interoperability and Integration Testing Framework. In *2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST)*, pages 454–459, April 2021. ISSN: 2159-4848.

[159] Qiaozhi Xu and Junxing Zhang. piFogBed: A Fog Computing Testbed Based on Raspberry Pi. In *2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC)*, pages 1–8, October 2019. ISSN: 2374-9628.

[160] Mingfu Li and Hung-Ju Lin. Design and Implementation of Smart Home Control Systems Based on Wireless Sensor Networks and Power Line Communications. *IEEE Transactions on Industrial Electronics*, 62(7):4430–4442, July 2015. Conference Name: IEEE Transactions on Industrial Electronics.

[161] Yao Ge, Shuja Ansari, Amir Abdulghani, Muhammad Ali Imran, and Qammer H. Abbasi. Intelligent Instruction-Based IoT Framework for Smart Home Applications using Speech Recognition. In *2020 IEEE International Conference on Smart Internet of Things (SmartIoT)*, pages 197–204, August 2020.

[162] Ehsan Nazerfard, Parisa Rashidi, and Diane J. Cook. Discovering Temporal Features and Relations of Activity Patterns. In *2010 IEEE International Conference on Data Mining Workshops*, pages 1069–1075, December 2010. ISSN: 2375-9259.

[163] Parisa Rashidi, Diane J. Cook, Lawrence B. Holder, and Maureen Schmitter-Edgecombe. Discovering Activities to Recognize and Track in a Smart Environment. *IEEE Transactions on Knowledge and Data Engineering*, 23(4):527–539, April 2011. Conference Name: IEEE Transactions on Knowledge and Data Engineering.

[164] Alli Mäkinen, Jaime Jiménez, and Roberto Morabito. ELIoT: Design of an emulated IoT platform. In *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pages 1–7, October 2017. ISSN: 2166-9589.

[165] Razvan Beuran, Lan Tien Nguyen, Toshiyuki Miyachi, Junya Nakata, Ken-ichi Chinen, Yasuo Tan, and Yoichi Shinoda. QOMB: A Wireless Network Emulation Testbed. In *GLOBECOM 2009 - 2009 IEEE Global Telecommunications Conference*, pages 1–6, November 2009. ISSN: 1930-529X.

[166] Chuong Dang Le Bao, Nhan Ly Trong, and Quan Le-Trung. UiTiOt: A Container-Based Network Emulation Testbed. In *Proceedings of the 2017 International Conference on Machine Learning and Soft Computing*, pages 161–166, Ho Chi Minh City Vietnam, January 2017. ACM.